Resum

Amb la realització d'aquest document, l'autor del projecte pretén mostrar als lectors els procediments seguits alhora de desenvolupar el treball de final de Màster, tant a nivell de la fase prèvia de definició i estudi del mercat com a nivell de desenvolupament del producte desenvolupat.

La finalitat del projecte és desenvolupar una aplicació per a tabletes electròniques que permeti interactuar amb una xarxa de comunicacions CAN (Controller Area Network), gràcies a la utilització del hardware USBtin que actua com a convertidor CAN/USB.

L'aplicació desenvolupada té la funció d'actuar com un *sniffer* de la xarxa CAN, per tal de que sigui capaç de mostrar el tràfic de dades que s'hi transmet a la vegada de dur a terme aquesta funció de dues formes diferents. D'una banda és capaç d'interactuar de forma directa amb la xarxa de comunicacions a través d'una interfície gràfica des de la tableta on està instal·lada. D'altra banda, l'aplicació també té desenvolupat un servidor web amb el que s'hi poden connectar diversos clients per accedir i manipular el hardware convertidor de forma remota.

Al llarg del document es recull informació tant referent al estudi previ realitzat per tal de definir els requeriments de l'aplicació, com els detalls de les fases de programació i desenvolupament tant de l'aplicació com del servidor web realitzades a través de la plataforma de creació d'aplicacions Android B4A.

A més a més, es mostra el resultat de les proves de validació del producte desenvolupat i es fa una estimació del pressupost del projecte a la vegada que es presenta i proposa unes línies d'investigació que poden ser executades en un futur en altres projectes.

Sumari

1. Glossari

B4A: Basic for Android

CAN: Controller Area Network

CiA: CAN in Automation

CSV: Comma Separated Values

DLC: Data Length Code

FIFO: First In First Out

HTML: HyperText Markup Language

HTTP: HyperText Transfer Protocol

IDE: Integrated Development Environment

IP: Internet Protocol

LAN: Local Area Network

URL: Uniform Resource Locator

USB: Universal Serial Bus

2. Introducció

2.1. Motivació

En els darrers anys ha explotat el món de les tecnologies mòbils i l'ús dels dispositius tals com els telèfons intel·ligents o *smartphones* i les tabletes electròniques. Aquest fet ha provocat que s'hagi creat un mercat ple d'oportunitats de negoci. Per exemple, es fa referència al negoci que es genera amb les xarxes socials i la publicitat associada, en l'oportunitat de comunicar-se amb altres persones de forma instantània que brinden aquests dispositius, la capacitat de consulta d'informació des de qualsevol lloc, entre moltes d'altres oportunitats.

Val a dir que totes aquestes funcionalitats que posen a disposició dels usuaris aquest tipus de dispositius tenen un denominador comú: les aplicacions o *apps*. El desenvolupament i la programació d'aquestes aplicacions ha suposat en la darrera dècada l'aparició de multituds d'oportunitats de negoci tant per a empreses multinacionals com per a persones particulars i joves emprenedors.

La idea del projecte neix en el moment en que l'autor encara l'etapa final dels estudis del Màster en Enginyeria Industrial MUEI a l'Escola Tècnica Superior d'Enginyeria Industrial de la UPC. En les experiències laborals prèvies a la realització d'aquest projecte, l'autor havia estat treballant en el món de l'automatització industrial amb la qual va adquirir coneixements respecte al protocol de comunicacions CAN.

El fet de que des del departament d'Enginyeria Electrònica de l'ETSEIB s'impulsa la realització de projectes relacionats amb aquest protocol i que en els darrers anys s'ha treballat amb la realització d'aplicacions per a dispositius mòbils per a comunicar-se amb una electrònica que incorpora un port de comunicacions CAN va fer que l'autor es mostrés atret amb l'idea de desenvolupar el projecte que es descriu en aquesta memòria.

2.2. Antecedents

En l'anterior apartat s'ha comentat que en els darrers anys des del Departament d'Enginyeria Electrònica de l'ETSEIB, de la mà del professor Manuel Moreno Eguílaz, s'ha impulsat la realització de projectes de desenvolupament d'aplicacions per a dispositius mòbils basades en el bus CAN.

Aquest fet és fruit a que el Departament d'Enginyeria Electrònica disposa d'una sèrie d'eines tant a nivell de hardware com de software idònies per al desenvolupament d'aquest tipus d'aplicacions. A nivell de hardware, el departament disposa d'una placa electrònica USBtin [1] de *low-cost* que actua com a un convertidor de protocol de comunicacions CAN a USB, entre d'altres.

A nivell de software el departament també disposa d'una plataforma de programació d'aplicacions per a dispositius amb sistema operatiu Android, conegut com Basic for Android B4A, que posa a disposició dels seus usuaris multitud d'enllaços informatius, tutorials, exemples de programació bàsics, llibreries d'usuari per al desenvolupament de tasques complexes, ...

En el seu dia, l'alumne de grau Marcel Garrido va realitzar un primer projecte enfocat a la utilització d'aquesta electrònica, desenvolupant una llibreria de funcions d'usuari per a B4A, que facilités la programació a altres usuaris que volguessin utilitzar el hardware USBtin [2]. Més endavant, es va impulsar un altre projecte desenvolupat per l'alumne de màster Héctor Hernández, en el que la llibreria implementada en l'anterior projecte va ser utilitzada, revisada i millorada per crear una aplicació capaç d'analitzar els cicles de conducció d'un vehicle [3].

Val a dir però que malgrat la realització d'aquests dos projectes anteriorment citats no s'ha implementat fins al moment l'eina típica de tots els protocols de comunicació com són els *sniffers*, que permeten als programadors copsar quina informació s'està transmetent per el bus de comunicacions.

Així doncs, el projecte del autor que es recull en aquesta memòria parteix d'aquests dos antecedents, ja que utilitza la llibreria implementada i revisada en el seu dia en els projectes citats en la bibliografia per programar una aplicació que sigui capaç de comunicar-se amb el hardware convertidor CAN/USB tant a través d'un dispositiu com pot ser una tableta, o bé de forma remota a través d'un servidor allotjat en el dispositiu Android.

2.3. Objectius

En qualsevol projecte es persegueix una sèrie d'objectius alhora d'executar-lo. És per això que abans d'arribar als capítols on s'explica els procediments emprats per fer realitat l'aplicació CAN Server en aquest punt es presenten els objectius marcats tant per l'autor com el tutor del projecte.

Com s'ha avançat en anteriors apartats l'objectiu principal que es persegueix és el desenvolupament d'una eina de diagnòstic que sigui capaç de copsar totes les dades que es transmeten a través d'una xarxa CAN, conegut popularment com una eina *sniffer*.

Per complir amb aquest objectiu s'opta per implementar una aplicació compatible amb sistemes operatius Android que es connecti a una xarxa de comunicacions CAN a través de la utilització d'un hardware.

Presentat l'objectiu principal del projecte, i essent aquest similar a moltes de les eines que existeixen en el mercat actual per diagnosticar l'estat d'un bus de comunicacions CAN és necessari plantejar un altre objectiu que permeti diferenciar l'aplicació CAN Server.

El nom de l'aplicació CAN Server està inspirat en el segon objectiu plantejat i es que es pretén que l'eina desenvolupada incorpori un servidor web que s'allotgi i es gestioni a través de l'aplicació i permeti la connexió a través d'un navegador web amb la xarxa de comunicacions CAN.

Així doncs, l'objectiu del projecte és dual, ja que es pretén desenvolupar una eina que pugui ser utilitzada a través de la tableta on estigui instal·lada l'aplicació CAN Server, com a través d'un navegador web accedint al servidor web.

A més a més d'aquests objectius de caràcter tècnic, es necessari plantejar altres metes a nivell formatiu al tractar-se d'un projecte de caire docent. Així doncs, l'autor del projecte es planteja l'objectiu d'aprendre un món desconegut per ell, com és el cas de la programació d'aplicacions Android com el desenvolupament de pàgines web en llenguatges de programació desconeguts per a ell. L'objectiu és aprendre de forma autònoma i amb l'ajut del tutor del projecte els coneixements necessaris per fer realitat l'aplicació CAN Server.

3. Estat de l'art

En qualsevol tipus de projecte és molt important dur a terme una fase d'estudi prèvia per determinar quins productes hi ha en el mercat actualment i que presentin unes característiques i/o prestacions similars al producte que es vol desenvolupar amb la realització d'aquest projecte.

El protocol de comunicacions CAN data els seus inicis en l'any 1986, quan l'empresa *Bosch* va llançar al mercat aquest mètode de transmissió de dades. Malgrat que hagin passat pràcticament 30 anys des del inici i hom pugui pensar que es tracta d'un bus industrialment obsolet, la forta implantació que va tenir en la industria de l'automoció, present en les xarxes de transmissió d'informació d'automòbils, camions, carrossers, sensors, actuadors, entre d'altres elements fa que actualment encara sigui una xarxa de comunicacions important a nivell industrial.

Així doncs, un protocol com aquest, que està fortament present en moltes aplicacions electròniques en les que es produeix transmissió de multitud de dades, ha d'anar acompanyat d'una sèrie d'eines de programació i validació. Com s'ha dit anteriorment en altres apartats, és comú que un protocol de comunicacions disposi d'un instrument que permeti saber quines dades s'estan transmetent per el bus, el que es coneix popularment com *sniffers*.

En el mercat, actualment ja existeixen molts analitzadors de xarxa desenvolupats per fabricants d'electrònica que posen a la disposició de desenvolupadors d'aplicacions industrials que incorporen comunicació industrial les eines necessàries per saber que és el que s'està enviant realment per el bus.

En aquest sentit doncs, el projecte de desenvolupament d'un *sniffer* CAN per una tableta electrònica no suposa una gran innovació, simplement destacar que la majoria d'aquests estan pensats per executar-los en PC's i no en sistemes operatius Android, com és el cas del producte que s'ha desenvolupat. Al llarg d'aquest apartat d'estudi de mercat es presenta algun dels *sniffers* CAN per a PC's que s'han consultat, com és el cas del *PCAN-View* del fabricant PEAK.

A nivell d'aplicacions que corren en sistema operatiu Android capaces d'emular el comportament dels *sniffers* que s'executen en un ordinador val a dir que les solucions existents al mercat són menors, és en aquest punt on entra en joc el producte que pretén desenvolupar-se amb aquest projecte i on es pretén diferenciar de la resta de projectes implementats els darrers anys per altres autors [2] i [3].

En definitiva, el producte que es vol desenvolupar es tracta d'un *sniffer* amb capacitat d'actuar i mostrar les dades que passen per el bus que està connectat al hardware USBtin, tant a nivell local des de la pròpia tableta electrònica, com a nivell remot permetent als usuaris que no estan connectats físicament al hardware veure quina informació s'hi està transmetent a través d'una pàgina web que la tableta emmagatzema i és capaç de servir als clients que s'hi connectin. És el punt fort que es vol afegir al mercat a través del desenvolupament d'aquest projecte i és que productes que presentin aquestes característiques ja no són tant comuns. En una segona part d'aquest punt de la memòria es fa referència a alguns dels productes que en són similars com és el cas d'un convertidor CAN-Bluetooth que corre en sistema operatiu Android desenvolupat per l'empresa GwentecEmbedded o bé un convertidor CAN-Wifi fabricat per ICPDAS-USA.

3.1. Analitzador de xarxa PCAN-View

L'empresa Peak Systems es basa en la fabricació d'electrònica capaç de comunicar-se mitjançant protocol de comunicació CAN. En la seva pàgina web es mostren una gran varietat de productes que van des de mòduls de posicionament GPS capaços de comunicar les seves dades via CAN, sondes de temperatura, mòduls d'entrades i sortides digitals, controladors i altres productes que comparteixen un denominador comú, la forma en que transmeten les dades es via CAN.

És per això que aquesta empresa ha desenvolupat una sèrie d'eines tant a nivell hardware com software que permeten comunicar-nos i mostrar el contingut de dades que s'envien per el bus. La solució hardware que presenten és similar a la que es disposa al Departament d'Electrònica de l'ETSEIB, una electrònica capaç de convertir les trames CAN a USB. L'empresa té a disposició diferents models d'aquest tipus de hardware que s'engloben dins de la família PCAN-USB. En la Figura 3-1 es pot veure algunes d'aquestes electròniques.

Figura 3-1. Convertidors CAN/USB de Peak Systems. Font [4]

Tots els productes PCAN-USB compleixen amb una sèrie d'especificacions, tals com que permeten comunicar-se amb diferents velocitats de transmissió CAN (des de 5kbit/s fins a 1Mbit/s, presenten un connector estàndard D-Sub de 9 pins, complint d'aquesta manera amb l'estàndard de la CiA [5], entre altres característiques tècniques de funcionament.

Amb la presentació d'aquesta família de productes PCAN-USB el que es vol transmetre al lector és que és l'equivalent a nivell de hardware de l'electrònica que s'utilitza en aquest projecte per desenvolupar l'aplicació. Val a dir però, que la gran diferència entre el hardware PCAN i el convertidor utilitzat en el projecte és el cost d'un producte o un altre, essent més assequible el convertidor USBtin. Per fer-nos una idea el model més econòmic de la família PCAN-USB té un PVP de 180,00€.

Fins al moment s'ha parlat de la solució hardware d'aquesta empresa, però aquesta ve acompanyada de la solució software que posa a disposició dels seus clients per tal de que a través d'aquesta eina es pugui analitzar el tràfic de dades del bus CAN, és a dir, l'eina *sniffer* de PEAK Systems. Aquest software, que ha desenvolupat l'empresa, és vàlid per els diferents productes de la gamma PCAN-USB i es coneix amb el nom de PCAN-View.

Figura 3-2.Aspecte de l'sniffer desenvolupat per l'empresa Peak Systems. Font: [4]

Les característiques principals d'aquesta eina són les següents:

I. Compliment de les especificacions CAN 2.0 A/B.
II. Transmissió manual i periòdica de missatges CAN amb una resolució màxima d'1ms.
III. Recepció de missatges CAN amb resolució màxima de 100 µs.
IV. Capacitat de representar els identificadors de missatge CAN-Id en format decimal i/o hexadecimal.
V. Representació dels bytes que contenen els missatges CAN en format ASCII, hexadecimal o decimal.
VI. Gravació de missatges per tenir un històric del tràfic de dades que circula per la xarxa CAN.
VII. Suporta la lectura de missatges amb ID's d'11 o 29 bits.
VIII. Permet dur a terme un reset del hardware PCAN-USB.

Val a dir que l'empresa Peak Systems proporciona aquesta eina de forma gratuïta conjuntament amb el hardware convertidor de CAN a USB. Aquest software està pensat per executar-lo en PCs i no en dispositius amb sistema operatiu Android, d'aquí sorgeix la principal diferència amb l'eina software desenvolupada en aquest projecte.

Tot i això, cal dir que a nivell de les prestacions i accions que ha de dur a terme l'aplicació Android a nivell local actuant com a *sniffer* el producte PCAN-View és un referent.

3.2. Convertidors CAN a *bluetooth*

Deixant de banda la solució basada en una electrònica convertidora de CAN-USB i una eina software que actuï com a *sniffer* i permeti visualitzar el tràfic d'una xarxa CAN en el mercat també es pot trobar diferents fabricants que utilitzen un hardware que converteix les trames d'informació CAN a comunicació *bluetooth,* amb els quals s'amplia la possibilitat de connexió més enllà dels ordinadors.

Un tipus de producte que respon a aquesta descripció que s'ha analitzat és l'eina desenvolupada per l'empresa GwentechEmbedded [6]. Com en el projecte actual divideix la solució en una part de hardware i una part de software, que en aquest cas es tracta d'una aplicació que es pot executar en sistemes operatius Android.

Per la part de hardware l'empresa ha fabricat una electrònica encapsulada que compleix amb els estàndards d'especificació CAN acordats per la CiA, la qual s'alimenta a través de tensió continua a 12V, i es connecta al bus CAN a través d'un connector D-sub 9 pins, de la mateixa forma que en el cas del PCAN-USB.

A més a més, com s'ha dit el propi fabricant ha desenvolupat una aplicació per Android denominada BluetoothCan Bus Analyzer, amb la qual es pot connectar-se a l'electrònica en qüestió i visualitzar les dades que s'estan enviant per el bus. L'aplicació permet l'enviament i la recepció de missatges CAN, establir diferents velocitats de transmissió del bus i saber la versió del firmware i hardware de l'electrònica.

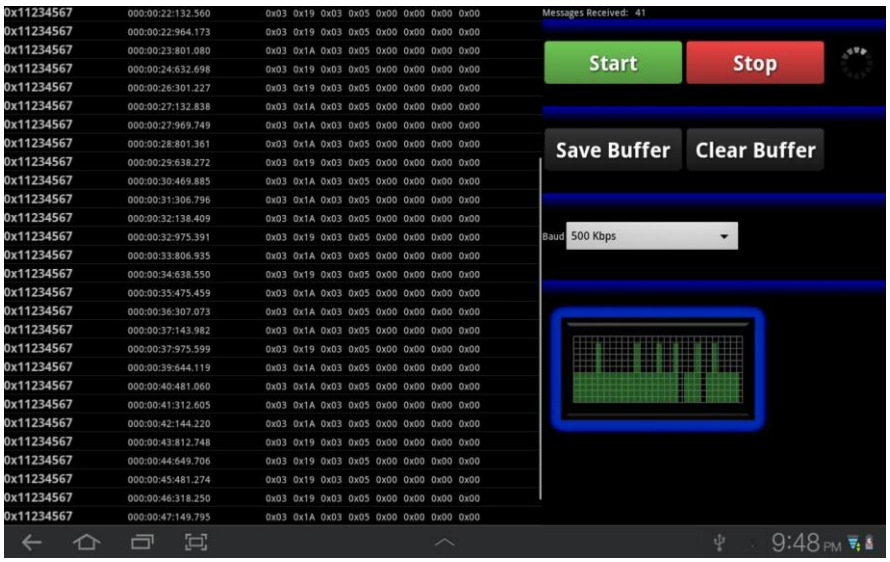


Figura 3-3. Captura de pantalla de l'APK Bluetooth CAN BUS Analyzer. Font [6]

Val a dir que a part de la solució proposada per l'empresa GwentechEmbedded, hi ha molts altres productes similars que utilitzen la solució CAN *to Bluetooth,* que es basa amb una electrònica que converteixi les trames de comunicació CAN a comunicació *Bluetooth*. Altres exemples existents en el mercat poden ser:

I. C2BT *(Can-BUS to BluetoothAdapter)* de l'empresa alemanaCaseGmbH.
II. CAN-BUS-*Bluetooth Converter* de l'empresa Matrix
III. CAN BUS to *BluetoothAdapter*BTS5904C2P de l'empresa OEM Blue.
IV. ...

El denominador comú de tots aquests productes es troba en la conversió que realitzen de les trames de comunicació CAN, que a diferència dels productes desenvolupats per l'empresa Peak Systems converteixen la informació a tecnologia *bluetooth* permetent d'aquesta forma que les eines software de testes puguin desenvolupar en multitud de sistemes operatius diferents, i és en aquest tipus de producte on es veu que hi ha aplicacions Android al mercat que s'utilitzen com analitzadors de xarxes CAN.

El producte desenvolupat en el projecte doncs utilitzarà una barreja de les opcions presentades fins al moment en aquests dos primers apartats, incorporant una solució hardware *low-cost* que converteixi les trames CAN a comunicació USB i una solució software a nivell d'aplicació Android, com pot ser el cas de l'aplicació Bluetooth CAN Bus Analyzer.

3.3. Convertidors CAN Wi-fi

Existeix una tercera solució a nivell tecnològic molt similar a les anteriors, però amb la diferenciació de que un equip electrònic és l'encarregat de convertir les trames de comunicació CAN a comunicació Wi-fi permetent que des d'un mòbil o tableta electrònica es pugui establir connexió amb el hardware convertidor i alhora aquest hardware estigui connectat físicament a una xarxa CAN.

És a dir, es tracta d'un producte similar al convertidor CANa Bluetooth o bé CAN a USB, però permet incorporar comunicació TCP i agilitzar i ampliar les possibilitats de connectivitat amb el bus de comunicacions CAN. L'empresa americana ICPDAS-USA ha desenvolupat i llançat al mercat un producte d'aquestes característiques anomenat I-7540D-WF [7].

El hardware I-7540D-WF es tracta d'una electrònica capaç de suportar la transmissió de dades d'una o múltiples xarxes CAN a través d'una xarxa WLAN d'acord amb el estàndard 802.11b/g d'aquest tipus de xarxes. Es tracta d'un dispositiu pensat per a que l'usuari es pugui connectar a una xarxa CAN de forma sense fils a xarxes mòbils de CAN (per exemple en automòbils o camions) o bé en xarxes CAN estacionaries.

Figura 3-4.Exemple d'aplicació del producte I-7540D-WF. Font [7]

Aquest producte té dos formes de funcionament, una actuant com un punt d'accés (AP) en el que es permet integrar aquesta electrònica dins d'una xarxa on existeixin diferents punts d'accés WLAN i es produeixi l'intercanvi d'informació entre aquests o bé el mètode *ad-hoc* o de connexió directa que permet connectar el I-7540D-WF amb un ordinador i així a través d'una eina software poder veure el tràfic de dades del bus CAN, com en els anteriors casos, o bé comunicar-se amb un segon I-7540D-WF, permetent comunicar dues xarxes CAN difícils d'unir físicament per cable.

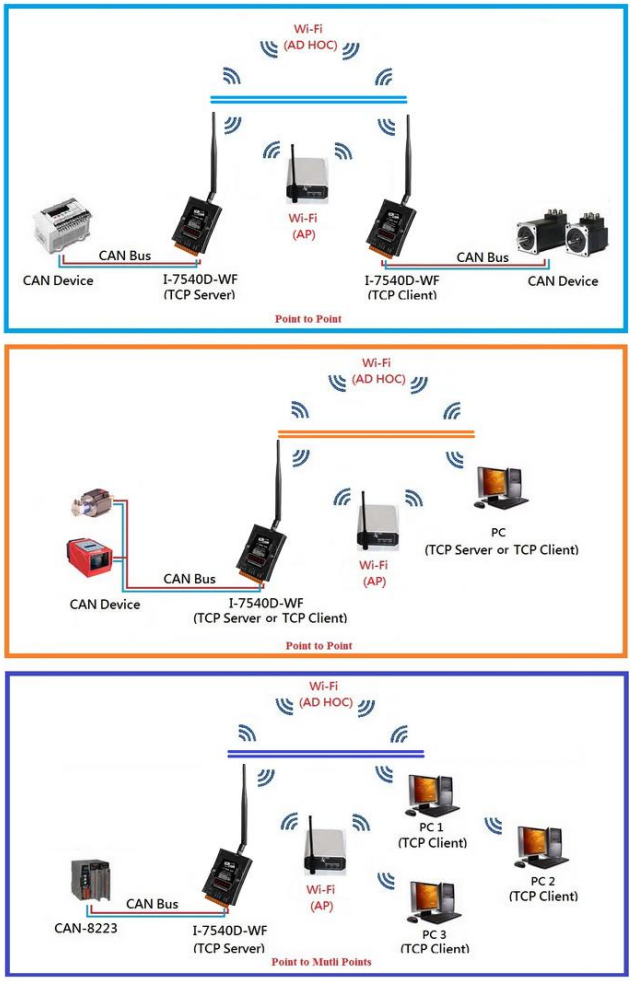


Figura 3-5. Esquema de funcionament del mode Ad-hoc. Font [7]

Val a dir que el producte en qüestió desenvolupat per l'empresa ICPDAS es focalitza en la solució hardware principalment, ja que a nivell de software ha desenvolupat l'eina *Utility,* que es tracta d'un *sniffer* que s'executa en un ordinador amb un aspecte i unes prestacions similars a les del PCAN-View vist en l'apartat 3.1.

És important destacar que el producte I-7540D-WF presenta un PVP a la pàgina web del fabricant de 699$. És en aquest aspecte, en l'econòmic on es vol diferenciar-se de la solució proposada per l'autor del projecte. A més a més d'implementar una solució software mitjançant la programació de l'aplicació CAN Server.

3.4. Anàlisi global de l'estat de l'art

Un cop vistos els diferents productes i solucions d'alguns fabricants respecte a la temàtica referent al diagnòstic i connectivitat a xarxes CAN per poder analitzar el tràfic d'informació que s'hi transmet és necessari fer un resum previ abans de definir les especificacions i el tipus de producte que es vol desenvolupar amb la realització d'aquest projecte.

Cal destacar que totes les solucions estudiades presenten dues parts ben diferenciades l'una de l'altra, com és el cas de la part electrònica i el relatiu al hardware que es connectarà a la xarxa i la forma amb que transmetrà les dades, sigui via protocol USB, *bluetooth, Wi-Fi,* etc... i una altra part enfocada a la programació d'una eina software de diagnòstic per a que l'usuari vegi el que està succeint a la xarxa CAN.

Cal destacar que totes les solucions estudiades presenten dues parts ben diferenciades l'una de l'altra, com és el cas de la part electrònica i el relatiu al hardware que es connectarà a la xarxa i la forma amb que transmetrà les dades, sigui via protocol USB, *bluetooth, Wi-Fi,* etc... i una altra part enfocada a la programació d'una eina software de diagnòstic per a que l'usuari vegi el que està succeint a la xarxa CAN.

D'altra banda, després d'analitzar els diferent productes al mercat cal extreure'n una altra conclusió i és que a nivell d'*sniffers* o software capaç d'analitzar quines dades s'estan transmetent per el bus val a dir que n'hi ha milers d'opcions disponibles al mercat, però que pràcticament la totalitat d'aquestes solucions estan pensades per ser executades des d'un ordinador personal, estan físicament connectats amb el hardware i la xarxa.

En el moment d'analitzar productes que siguin capaços de dur a terme la funció *sniffer* de forma sense fils o des d'altres dispositius mòbils que no siguin un PC el nombre de solucions baixa considerablement i el preu d'aquestes solucions no és *low-cost*.

4. Requeriments i especificacions

En tot projecte és important fer una definició prèvia dels requeriments i especificacions que es volen complir amb la seva execució. És en aquest punt de la memòria on es defineixen totes les funcionalitats que l'autor ha implementat en l'aplicació Android desenvolupada.

Com s'ha vingut avançant en anteriors apartats el producte que es pretén desenvolupar és un *sniffer* CAN, és a dir, una eina capaç de monitoritzar el tràfic de dades d'una xarxa de comunicacions CAN. Aquest tipus de productes, tot i presentar un aspecte similar els uns dels altres poden arribar a presentar variacions i oferir prestacions diferents. La principal diferenciació que vol aconseguir l'autor del projecte respecte altres productes similars és la de donar l'opció als usuaris de comunicar-se de forma remota amb una xarxa que inclogui el protocol CAN, sense la necessitat d'estar connectat físicament al dispositiu electrònic de la xarxa.

Per tal d'aconseguir aquesta prestació s'optarà per el desenvolupament d'una arquitectura basada en que l'aplicació allotjarà un servidor web en el qual s'hi podran connectar diferents clients/usuaris per consultar la informació que s'està transmetent per la xarxa CAN, o bé adquirir fitxers d'històrics de dades sense la necessitat d'estar físicament al lloc on està instal·lada la xarxa de comunicació.

Amb aquesta idea doncs, es divideix el projecte i el desenvolupament de l'aplicació en dues parts diferenciades, una part local en la que es permetrà interactuar amb la xarxa des del dispositiu on s'instal·li l'aplicació, i una part que inclogui el desenvolupament d'un servidor web que permeti interactuar amb l'aplicació de forma remota.

És per això que en aquest apartat també es diferenciarà entre les funcionalitats que s'implementin a nivell local, i les que es facin a nivell remot.

4.1. Requisits de funcionament a nivell local

Es tracta de crear una interfície gràfica que es mostri quan s'executi l'aplicació i que permeti al usuari treballar directament amb la targeta convertidora USBtin, utilitzant en bona mesura les funcions disponibles de la llibreria CANBUS [2]. Per això els punts que ha d'acomplir l'aplicació són els següents:

I. Gestió de la connexió amb el hardware USBtin

Es creu convenient que aquest procediment de connexió amb el hardware sigui automàtic un cop s'iniciï l'aplicació i que l'usuari tan sols es preocupi de dur a terme la connexió física de la targeta USBtin al port USB de la tableta.

- Connectar-se mitjançant el port USB amb el hardware i establir comunicació entre l'aplicació i la targeta USBtin.
- Crear un botó que permeti reiniciar el procediment de connexió mitjançant el port USB per resoldre les situacions en les que es perdi la comunicació USB.
- Crear un visor que mostri l'estat de la comunicació amb la targeta i informi a l'usuari si està connectat al hardware o no.

II. Gestió de l'estat del port CAN del hardware USBtin

Es tracta de donar la possibilitat al usuari d'obrir i tancar el port CAN quan li convingui, i poder establir una velocitat de transmissió concreta.

- Crear un selector que sigui un menú desplegable amb les diferents velocitats de transmissió (baudrates del bus CAN) disponibles per donar l'opció d'escollir la velocitat desitjada per l'usuari.
- Crear un botó que sigui l'encarregat d'obrir el port CAN del hardware a la velocitat determinada per el selector.
- Crear un botó que tanqui el port CAN
- Crear un visor tipus semàfor que mostri l'estat del port CAN, essent el seu color verd quan el port està obert i vermell quan el port CAN està tancat.

III. Gestió de la informació obtinguda a través de la targeta USBtin

Es tracta de que l'aplicació sigui capaç d'adquirir el tràfic de dades de la xarxa de comunicacions CAN que la targeta USBtin capta i gestionar la informació per mostrar-la als usuaris.

- Crear una taula que reculli els missatges CAN rebuts i mostri les dades a través de la interfície gràfica de la tableta, definint una cua de missatges (FIFO) que mostri els últims missatges rebuts.
- Generar un fitxer que emmagatzemi tots els missatges rebuts i guardar-lo en la memòria interna del dispositiu on s'instal·li l'aplicació.
- Crear un botó que permeti al usuari decidir si es vol dur a terme la creació de fitxers que recullin l'històric de missatges rebuts en l'aplicació.

4.2. Requisits de funcionament del servidor web

Es tracta de crear un seguit de pàgines web que s'allotjaran en l'aplicació quan s'instal·li en una tableta i crear un servei que corri de forma paral·lela a l'aplicació per tal de generar un servidor web al que es puguin connectar diferents clients connectats a la mateixa xarxa a la que està connectada la tableta. L'objectiu principal d'aquesta part es reproduir les funcionalitats de nivell local i permetre que els clients que es connectin al servidor puguin interactuar amb la targeta USBtin.

I. Gestió de l'accés al servidor web

Aquesta part contempla la part d'accés al servidor web, el procediment que han de seguir els usuaris que s'hi vulguin connectar.

- Assignar una direcció IP concreta amb un port d'accés específic (5555) per tal de que els clients accedeixin a la pàgina web de credencials (*login)* del servidor des de qualsevol navegador web. Cal destacar que s'ha d'estar connectat dins de la mateixa xarxa que la tableta on estigui instal·lada l'aplicació.
- Crear una pàgina web que incorpori un nivell de seguretat mínim per demanar el nom d'usuari i contrasenya als clients que es connectin al servidor.
- Crear un botó en la pàgina web que permeti accedir a la pàgina web principal si s'han introduït correctament les credencials o mostri un *pop-up* d'avís si aquestes són incorrectes.

II. Visualització de l'estat de la comunicació USB amb el dispositiu USBtin

Aquesta part bàsicament consisteix en informar als clients que es connectin al servidor web de si el hardware USBtin està connectat a l'aplicació i per tant es pot treballar amb ell o no.

- Crear un visor en la pàgina principal del servidor web que mostri als usuaris si el hardware està connectat o no.

III. Gestió de l'estat del port CAN de forma remota

En aquest punt es tracta de reproduir les funcionalitats a nivell local d'aquest ítem. Poder obrir i tancar el port CAN a una velocitat de transmissió concreta a través del servidor web.

- Crear un menú desplegable en la pàgina web principal amb les diferents opcions disponibles de velocitat de transmissió.
- Crear un botó que s'encarregui d'obrir el port CAN a la velocitat definida en el menú desplegable
- Crear un botó que tanqui el port CAN.

- Generar uns visors que mostrin l'estat del port CAN i la velocitat de transmissió a la que s'està treballant.

IV. Gestió dels missatges CAN rebuts a la targeta USBtin

Es tracta de gestionar el tràfic de la xarxa de comunicacions CAN i que l'usuari pugui consultar quins missatges s'han transmès per la xarxa.

- Crear una taula que reculli de forma *online* els missatges que s'estan transmetent a la xarxa de comunicacions CAN des dels servidor web.
- Crear una pàgina web en que es mostrin tots els fitxers d'històrics creats per l'aplicació i que es puguin descarregar des del servidor web.
- Permetre visualitzar el contingut dels fitxers històrics en la pròpia pàgina web prèviament a ser descarregats pels clients.

5. Eines de desenvolupament

Per tal de progressar en l'execució del projecte, i en conseqüència el desenvolupament de l'aplicació especificada per l'autor, és necessari la utilització d'una sèrie d'eines. En l'apartat *3.Estat de l'art* d'aquesta memòria s'ha vist com la majoria de fabricants de solucions per al test d'aplicacions on hi ha la presència d'una xarxa de comunicacions CAN desenvolupen els seus productes mitjançant eines que estan basades en la combinació de solucions software i hardware.

Així doncs, l'autor del projecte, per desenvolupar l'aplicació que compleixi amb les especificacions marcades, ha combinat la utilització de diferents eines software (a nivell de programació de l'aplicació i de la pàgina web) i del hardware USBtin, que es presentaran en aquest punt de la memòria i que són les següents:

- ✓ Plataforma B4A
- ✓ Editor de text *Sublime Text* HTML
- ✓ Hardware USBtin convertidor CAN a USB

Figura 5-1. Logotips de les diferents eines de desenvolupament utilitzades. Font pròpia

L'elecció d'aquestes eines va venir gràcies a que des del Departament d'Electrònica de l'ETSEIB es disposa d'una llicència del software B4A i aquest ja va ser utilitzat en anteriors projectes impulsats pel Departament, com és el cas dels projectes citats en l'apartat *2.2. Antecedents* [2] i [3].

Gràcies al desenvolupament d'una llibreria de funcions dedicades a la comunicació amb el hardware USBtin desenvolupada en aquests projectes es creu que les eines USBtin i B4A són les més òptimes des d'un punt de vista de temps de desenvolupament per la programació de l'aplicació Android.

De cara al desenvolupament de la pàgina web es creu que el llenguatge HTML, molt estandarditzat en el món de la programació web pot ser una eina útil gràcies a la quantitat d'informació i tutorials presents a Internet. Per la seva programació, tot i que podria utilitzar-se qualsevol editor de text, s'escull l'editor de text *Sublime Text,* ja que l'autor ja el coneixia amb anterioritat.

5.1. Plataforma B4A

La gran majoria d'aplicacions que es desenvolupen avui en dia estan programades des del entorn *Android Studio,* que és el IDE oficial per al desenvolupament d'aplicacions que corren sobre sistemes operatius Android amb llenguatge de programació Java. Malgrat això, com s'ha avançat en l'anterior punt, l'eina escollida per el desenvolupament de l'aplicació Android CAN-Server és un software de programació anomenat *Basic for Android,*desenvolupat per l'empresa *Anywhere Software.*

Es tracta d'una plataforma software que no utilitza el llenguatge de programació Java, sinó que n'utilitza un llenguatge de programació propi, tot i que manté la filosofia de programació orientada a objectes i esdeveniments de Java. És molt similar al llenguatge Visual Basic o al .NET.

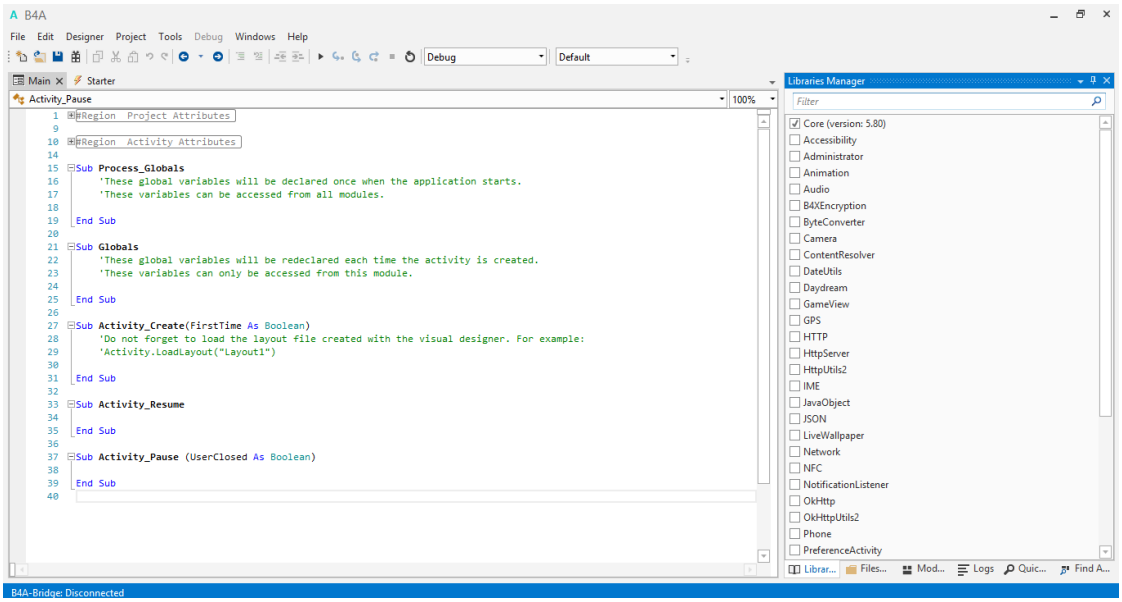


Figura 5-2. Captura de pantalla de l'entorn de programació B4A. Font pròpia

A més a més, B4A posa a disposició dels programadors una eina de disseny visual amb la qual es pot dur a terme el disseny gràfic de l'aplicació distribuint visors, botons, imatges i multitud de figures gràfiques, per tal de maquetar l'aplicació mitjançant les opcions de disseny de colors, formes, visibilitat, format de text, etc ... D'aquesta forma també es permet adaptar el disseny visual a les diferents mesures i dimensions de diferents models de mòbils i tabletes electròniques.

Val a dir que totes aquestes tasques de disseny poden ser executades a través de codi i programació per part del desenvolupador, però mitjançant el B4A Visual Studio es redueix la complexitat de la programació a nivell de disseny visual.

A més a més, la plataforma també inclou una opció de depuració (*debbuging)* per tal de poder depurar i testejar la programació de l'aplicació. Es permet aquesta opció gràcies a l'aplicació B4A Bridge, que es pot descarregar des del *Play Store* i que enllaça l'ordinador des d'on s'està desenvolupant l'aplicació amb el B4A i el mòbil o dispositiu al que es vol fer córrer l'aplicació.

D'aquesta forma, gràcies a l'enllaç que es crea, s'instal·la l'aplicació al dispositiu i es pot dur a terme l'execució del codi pas a pas, incloure punts de ruptura o *break-points* d'execució per testejar i validar parts específiques del codi i observar possibles problemes en la programació per resoldre'ls, a més de donar una visió al programador del que s'està executant.

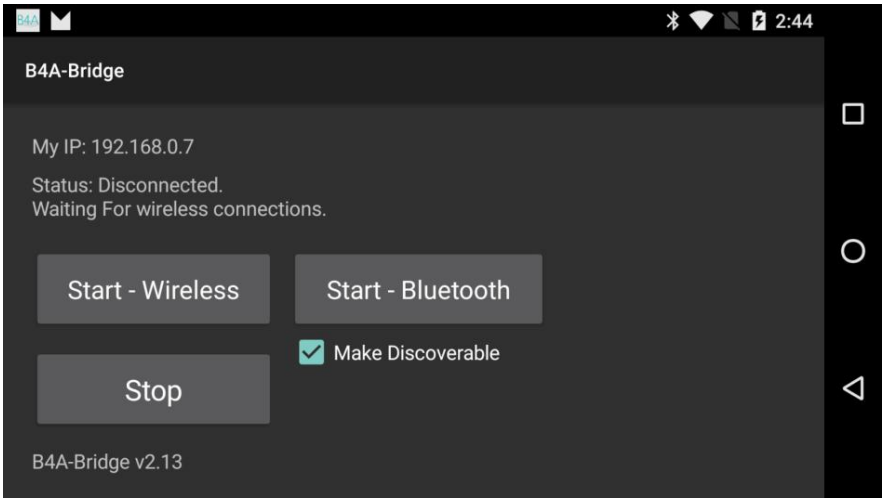


Figura 5-3. Captura de pantalla de l'aplicació B4A-Bridge que enllaça el dispositiu i el PC. Font pròpia

Com es pot veure en la *Figura 5-3*, l'aplicació B4A-Bridge es senzilla i per enllaçar el PC amb el dispositiu tan sols fa falta que tots dos estiguin connectats en la mateixa xarxa Wi-Fi, i introduïnt l'adreça IP que ens facilita B4A-Bridge a l'opció que dóna el software B4A es pot descarregar l'aplicació al dispositiu i fer una depuració del codi a temps real.

Un dels altres factors que va resultar ser decisiu en l'elecció d'aquesta plataforma software és que existeix una comunitat amb més de 75.000 programadors registrats i posa a la disposició dels seus usuaris un fòrum on poder plantejar dubtes, veure exemples i tutorials, descarregar-se llibreries de funcions dedicades, i des d'on els treballadors de l'empresa *Anywhere Software* donen suport als membres registrats. [8]

Cal dir que B4A és una plataforma de pagament i que posa a disposició dels usuaris una versió *demo* de 30 dies que un cop exhaurida s'ha de comprar una llicència per la seva utilització i que té un PVP de 119$.

5.2. Hardware convertidor USBtin

Per tal de dur a terme la connexió física amb una xarxa de comunicacions que utilitzi el protocol CAN és necessari un dispositiu electrònic. Val a dir que per tal de poder tirar endavant el projecte era necessari definir un hardware específic capaç de connectar-se en aquest tipus de xarxa i poder d'obtenir la informació que s'hi esta transmetent.

Com s'ha vist en l'apartat *3. Estat de l'art* hi ha multitud de dispositius electrònics amb capacitat per dur a terme aquestes tasques, però ja que la idea del projecte és desenvolupar una aplicació *low-cost* es creu convenient que el hardware que s'utilitzi sigui una electrònica *open-source* i amb un cost econòmic raonable. Aquesta descripció encaixa perfectament amb les característiques d'un convertidor CAN a USB que disposa el Departament d'Electrònica de l'ETSEIB i el fet de que fos utilitzat en projectes anteriors va resultar determinant per elegir aquest hardware.

Es tracta d'una placa electrònica desenvolupada per Thomas Fischl [1] de caràcter *open-source,* ja que en la seva pàgina web posa a disposició dels usuaris tota la informació del dispositiu, incloent el *datasheet,* l'esquemàtic de la placa, la referència dels components que la integrin per si els usuaris volen dissenyar el seu propi USBtin, i tota la informació referent al procés de comunicació de l'aparell amb la xarxa CAN.

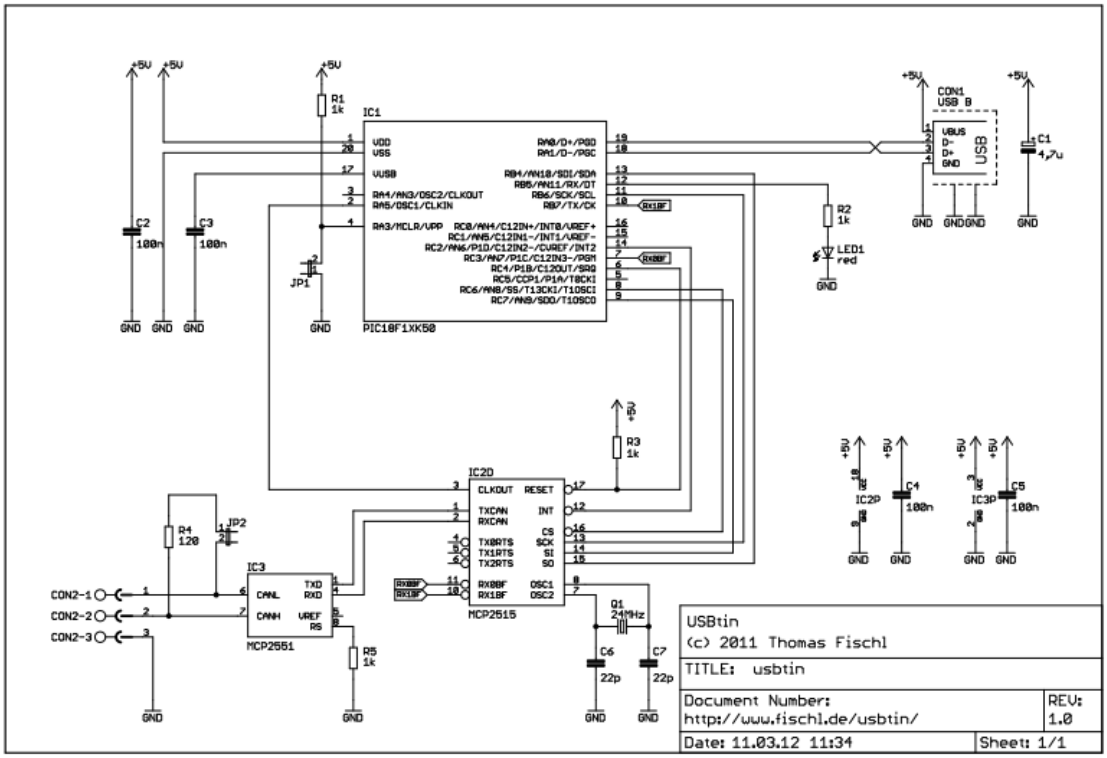


Figura 5-4. Circuit esquemàtic de la placa USBtin. Font [1]

A més a més, des de la pàgina web del producte també es pot consultar els comandos que s'han d'enviar al dispositiu per tal de dur a terme una acció concreta, com pot ser establir la velocitat de transmissió de dades o *baudrate*, l'enviament o lectura d'un missatge CAN entre d'altres accions.

Els usuaris també poden descarregar-se des de la pàgina web del desenvolupador d'aquest hardware diferents eines per testejar l'aparell, pensades sobretot per utilitzar el hardware com a *sniffer* d'una xarxa de comunicacions CAN.

5.3. Llenguatge HTML

Per tal de poder desenvolupar la funcionalitat de *webserver* amb la que poder connectar-se a la tableta electrònica des de qualsevol navegador web connectat a la mateixa xarxa LAN que el dispositiu on està instal·lada l'aplicació és necessari desenvolupar una pàgina web que mostri a l'usuari les opcions que té d'actuar sobre el dispositiu hardware de forma remota.

Val a dir que aquesta part del projecte forma part del que l'autor anomena com a funcionalitat remota, i es que l'aplicació que s'instal·larà en la tableta electrònica allotjarà una pàgina web per tal de que quan qualsevol client es connecti a l'adreça d'aquest dispositiu l'hi mostri una pàgina web amb diferents opcions per interactuar amb el hardware USBtin i visualitzar que està passant a la xarxa CAN.

Un dels llenguatges de programació web més estes i més popular es tracta del llenguatge HTML, el qual es pot generar des de qualsevol editor de text, i executant-ho en un navegador permet la visualització de la pàgina web programada. Així doncs, utilitzant l'editor de text *SublimeText,* l'autor ha desenvolupat la part de la pàgina web que inclou el projecte.

Per la familiarització de l'autor amb aquest tipus de programació s'ha consultat diferents tutorials i pàgines web on es mostren exemples i documents d'iniciació a la programació web [9].

6. Estructura de l'aplicació

Arribats en aquest apartat de la memòria el que es pretén en aquest punt és descriure i explicar l'arquitectura de la aplicació CAN-Server programada a través de la plataforma B4A per donar una visió global al lector de com s'ha estructurat el codi de programació, quines rutines s'utilitzen per el funcionament a nivell local i quines es fan servir per el funcionament remot, veure les diferents llibreries utilitzades i com han estat integrades en l'aplicació.

En un primer apartat es presentarà les diferents parts que conformen l'aplicació per tal de que el lector pugui entendre la seva estructura, i com ja s'ha fet en anteriors apartats es prosseguirà a continuació amb un altre apartat que desenvoluparà l'explicació de les funcionalitats locals i un altre punt en que l'autor parla de com s'ha programat la part de funcionalitats remotes.

6.1. Arquitectura de l'aplicació

L'aplicació CAN Server està composta per diferents mòduls i classes que al seu torn engloben una sèrie de variables específiques, rutines encarregades de dur a terme una funció específica o inclouen la gestió de serveis, com pot ser el cas de la gestió del servidor web.

El fet d'haver d'implementar una sèrie de funcionalitats i gestionar diversos esdeveniments i processos fa que el codi de l'aplicació ocupi un volum força elevat. És per aquest motiu, que l'aplicació s'estructura en diferents mòduls per poder tenir un control i que el codi sigui el més entenedor possible.

Bàsicament el contingut de l'aplicació es divideix en els següents elements que es poden veure de forma visual en la figura Figura 6-1.

- ✓ Rutina principal d'execució *Main* que s'encarrega de gestionar l'aplicació a nivell local, dur a terme els processos d'inicialització pertinents, cridar les funcions de les llibreries, controlar els esdeveniments que es generen de la interacció de l'usuari amb els elements visuals de l'aplicació entre altres funcions.

- ✓ Servei que controla la gestió del servidor web. S'inicialitza conjuntament amb l'aplicació i si no s'atura a través d'un botó des de la tableta s'executa de forma continuada. La seva funció es crear el servidor web, allotjar les diferents pàgines web i atendre les peticions del clients que es connecten al servidor.

- ✓ Fitxer de visualització generat amb l'eina B4A Designer que posa a disposició la plataforma B4A. Conté la part gràfica i la que es mostrarà pel dispositiu quan s'utilitzi l'aplicació. Els elements que conté, com poden ser botons, visors, llistes entre d'altres són controlats des de la rutina principal.

- ✓ Mòduls BAS o llibreries d'usuari. A més a més, de les llibreries que s'incorporen de forma automàtica i que estan disponibles a la plataforma B4A s'incorpora a l'aplicació la llibreria CANBUS [2] i Table *[10]*, que es tracta de codi estructurat en diferents rutines que poden ser cridades i executades des de la rutina principal Main per controlar el hardware USBtin.

- ✓ Els fitxers HTML són fitxers que contenen el codi font de les pàgines web i que han estat programats en paral·lel amb el codi de l'aplicació B4A ja que des del servei *Server Service* es carreguen aquests fitxers per tal de mostrar-los des del servidor web quan els clients que es connectin ho sol·licitin.

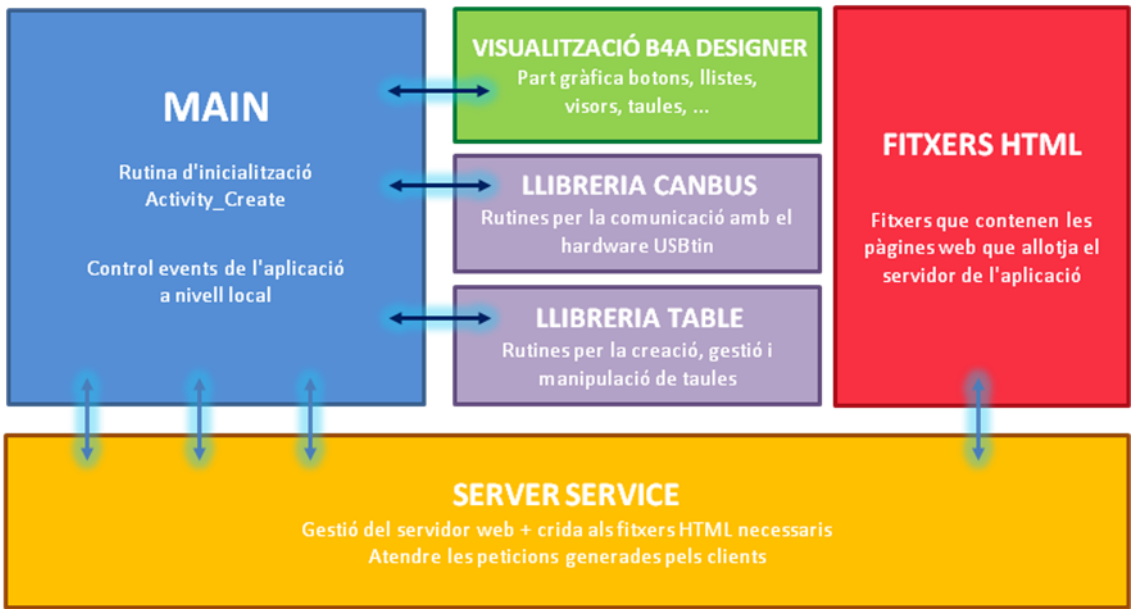


Figura 6-1. Diagrama de blocs que conté els diferents mòduls de l'aplicació CAN Server. Font pròpia

6.2. Aplicació a nivell local

Abans d'entrar en matèria sobre l'explicació de com s'ha procedit en la programació de les diferents funcionalitats de l'aplicació a nivell local per a que l'usuari interactuï directament amb la xarxa de comunicacions CAN des del dispositiu on s'instal·la l'aplicació és bo llistar els punts que afecten en aquest apartat:

- ✓ Comunicació amb el hardware USBtin a través del port USB.
- ✓ Configuració de la connexió amb el port CAN i gestió de la velocitat de transmissió.
- ✓ Lectura del tràfic d'informació en forma de missatges que es transmeten per la xarxa de comunicacions.
- ✓ Gestió de la informació mitjançant la creació d'històrics de dades que emmagatzemin les dades capturades.

Aquesta llista de punts que apliquen al funcionament local han estat desenvolupats en la rutina principal *Main* a través de la crida de funcions de la llibreria CANBUS (utilitzada en altres projectes [1] i [2] però que ha estat revisada per adaptar-la a les necessitats de l'autor) i una llibreria anomenada Table [10] amb la qual s'ha gestionat la manera en que es mostra la informació capturada per l'USBtin a través de la pantalla del dispositiu.

6.2.1. Comunicació entre l'aplicació i l'USBtin

El hardware USBtin basa el seu principi de funcionament amb la comunicació a través del port USB, de manera que si li arriben unes comandes concretes, (veure Annex 3) aquest actua en conseqüència i fa la conversió pertinent entre la informació que es transmet per protocol CAN a comunicació USB. Així doncs, l'aplicació requereix de l'establiment d'una comunicació USB amb el hardware.

Per tal de poder comunicar-nos amb el hardware USBtin i d'aquesta manera intercanviar comandes per governar el port CAN que incorpora i obtenir informació de la xarxa de comunicacions CAN a la que es connecti l'aparell cal establir una connexió física entre l'USBtin i el dispositiu electrònic on s'instal·li l'aplicació.

Com que es tracta d'un convertidor CAN a USB, aquesta connexió es duu a terme a través del port micro-USB tipus B que incorpora la tableta electrònica amb la que s'ha treballat. Per fer les connexions s'utilitza un adaptador de micro-USB tipus B a un connector USB i un cable amb connector mini-USB.

Figura 6-2. Esquema gràfic de les connexions entre tableta i USBtin. Font pròpia

Cal especificar que aquesta part de comunicacions USB s'executa de forma automàtica al iniciar l'aplicació, sense necessitat de que l'usuari premi botons, ja que sempre que es faci córrer l'aplicació tindrà l'objectiu final d'interactuar amb el hardware i d'aquesta forma se'n facilita l'ús. En definitiva, les comunicacions USB són transparents per a l'usuari.

Així doncs, dins de la rutina principal d'execució *Main*, en l'activitat que el sistema operatiu fa córrer a l'iniciar l'aplicació s'executen les funcions dedicades de la llibreria CANBUS per inicialitzar la comunicació amb l'USBtin, que bàsicament és obrir el port USB. En el primer moment d'execució es demana permisos d'administrador per tal de que l'aplicació pugui utilitzar aquest port i seguidament s'estableix la connexió mitjançant protocol de comunicacions USB amb el dispositiu. Cal destacar que la velocitat de transmissió de dades via USB ha de ser de 115200 bauds/s obligatòriament.

Un cop establerta la connexió des de la llibreria CANBUS s'inicialitzen dos búffers, un de recepció i un altre de transmissió, que gestionaran la transmissió de dades entre dispositius.

S'introdueix un visor en la pantalla del dispositiu Android per tal de mostrar l'estat de connexió amb el hardware USBtin, per avisar al usuari si s'ha establert o no la comunicació correctament.

```
'Inicializar la conexión con el hardware USBtin via puerto USB

'Pedir permiso para la utilizacion del puerto USB si no se tiene. Primera vez que se ejecuta la aplicacion!
If canbus1.usb1.HasPermission(1) = False Then
    canbus1.usb1.RequestPermission(1)
Else If canbus1.usb1.HasPermission(1) = True Then
    canbus1.usbConnect(BAUDRATE_USB)
End If
```

Figura 6-3. Procediment d'inicialització de la comunicació per el port USB. Font pròpia

6.2.2. Configuració port CAN

Per tal d'interactuar i poder implementar una eina *sniffer* és necessari configurar i obrir el port de comunicacions CAN que incorpora l'electrònica USBtin. Per aconseguir aquest punt la llibreria CANBUS presenta funcions dedicades a l'obertura i el tancament del port CAN, a més a més d'una funció especial per definir la velocitat de transmissió *baudrate* amb la que es connecta el dispositiu a la xarxa.

En les xarxes CAN és comú veure velocitats de transmissió diferents, i per tant es veu una necessitat de donar l'oportunitat a l'usuari de que sigui capaç d'escollir aquest paràmetre. Fins ara, els anteriors projectes citats en els antecedents [1] i [2] van ser desenvolupats per treballar sempre amb la mateixa velocitat de transmissió.

L'idea de desenvolupar una eina *sniffer* a una velocitat de transmissió fixa no té lògica ja que tan sols podria ser utilitzada en un determinat rang de *baudrates*. És per això que l'autor implementa la possibilitat de deixar escollir a l'usuari a quina velocitat de transmissió de les que permet el dispositiu USBtin vol treballar.

Per desenvolupar aquest punt es va crear un camp desplegable amb les diferents velocitats de transmissió disponibles. Aquest camp es va crear afegint l'element *Spinner* amb el nom *selector_BAUD* que es posa a disposició des de B4A, amb el qual es permet afegir ítems que després podran ser seleccionats en una llista que es mostra a l'aplicació en forma de menú desplegable.

A més a més, l'element *Spinner* crea una subrutina que retorna el valor seleccionat, el qual pot ser emmagatzemat en una variable. Així doncs, en l'aplicació hi ha una subrutina capaç de detectar quan s'ha polsat un element de la llista i emmagatzemar en una variable la posició que equival. D'aquesta forma es té la velocitat de transmissió que escull l'usuari en una variable.

```
'Selector de la velocidad de transmisión Baudrate de la comunicación CAN
Sub selector_BAUD_ItemClick (Position As Int, Value As Object)
 If selector_BAUD.SelectedIndex=0 Then BAUDRATE_CAN=0
 If selector_BAUD.SelectedIndex=1 Then BAUDRATE_CAN=1
 If selector_BAUD.SelectedIndex=2 Then BAUDRATE_CAN=2
 If selector_BAUD.SelectedIndex=3 Then BAUDRATE_CAN=3
 If selector_BAUD.SelectedIndex=4 Then BAUDRATE_CAN=4
 If selector_BAUD.SelectedIndex=5 Then BAUDRATE_CAN=5
 If selector_BAUD.SelectedIndex=6 Then BAUDRATE_CAN=6
 If selector_BAUD.SelectedIndex=7 Then BAUDRATE_CAN=7
 If selector_BAUD.SelectedIndex=7 Then BAUDRATE_CAN=8
 Log(BAUDRATE_CAN)
End Sub
```

Figura 6-4. Subrutina que s'implementa per recollir la velocitat de transmissió de la xarxa CAN a la que es vol connectar. Font pròpia

Val a dir però que el punt referent a la possibilitat d'escollir la velocitat de transmissió de la xarxa CAN a la que es vol connectar genera una tasca paral·lela per al seu compliment.

El dispositiu USBtin, com la major part del hardware amb ports de comunicació CAN, necessita reiniciar el port CAN per tal de fer efectiu un canvi de la velocitat de transmissió. Així doncs, si l'usuari que executi l'aplicació vol treballar a diferents velocitats de transmissió ha de tenir la possibilitat de tancar i obrir el port CAN.

És per aquest motiu que en aquest punt es va definir la creació de dos botons que executessin dues rutines, una encarregada d'obrir el port CAN al *baudrate* escollit i una altra subrutina destinada a tancar aquest port del dispositiu. Aquestes subrutines són executades utilitzant l'esdeveniment *click* que incorporen els botons afegits.

```
'Botón que ejecuta Sub OpenCAN
Sub btn_OpenCAN_Click
    OpenCAN(BAUDRATE_CAN)
End Sub

'Botón que ejecuta Sub CloseCAN
Sub btn_CloseCAN_Click
    CloseCAN
End Sub
```

Figura 6-5. Crida a les funcions dedicades al control del port CAN mitjançant el clic del botons. Font pròpia

A més a més, ja que el dispositiu no accepta canvis de velocitat *"online"*, és a dir, amb el port CAN obert, s'inhabilita des de l'aplicació el menú desplegable del selector de *baudrate* de manera que l'usuari no pugui canviar la velocitat de transmissió, ja que si ho fes, el canvi no es faria efectiu.

Per tal d'evitar funcionaments inadequats per part de l'usuari, tals com pot ser executar la rutina d'obertura del port CAN, un cop aquest està obert, o de forma inversa estar constantment tancant el port, tot i que aquest ja ho estigui, l'autor del projecte va implementar una lògica d'habilitació i inhabilitació d'aquests botons en funció de l'estat del port CAN, a més d'incloure un visor tipus semàfor que indica l'estat del port, essent el color verd l'indicador de que el port està obert i el vermell l'indicador de que està tancat.

6.2.3. Visualització del tràfic de dades de la xarxa de comunicacions CAN

Tal i com s'ha anat comentant al llarg de la memòria, l'objectiu principal dels *sniffers* és la de donar la visió del que està passant al bus de comunicacions, és a dir, mostrar tota la informació que s'està transmetent en aquest cas per una xarxa de comunicacions amb protocol CAN.

Per poder dur a terme aquesta funcionalitat és necessari poder llegir els missatges que circulen per la xarxa i mostrar-ho per l'aplicació per tal de que l'usuari pugui visualitzar aquests missatges a nivell local. L'idea del autor del projecte és mostrar per la pantalla del dispositiu on s'executa l'aplicació els missatges que es van llegint.

El protocol de CAN es basa en la transmissió de la informació a través de missatges amb un identificador i una longitud determinada de fins un màxim de 8 bytes. Aquests missatges presenten una estructura marcada que consta de l'identificador de missatge que pot ser curt (11 bits) o extens (29 bits), el DLC que indica la longitud del missatge, i els propis bytes d'informació.

Al tractar-se d'una xarxa de comunicacions amb tràfic de missatges es planteja l'opció de que un cop obert el port CAN l'aplicació està constantment llegint i capturant els missatges que es transmeten i tracta la informació per tal de mostrar-la de forma ordenada i entenedora en l'aplicació. Es creu convenient utilitzar el format taula per recollir els diferents camps que conformen un missatge CAN a més a més d'incorporar la data i hora en que s'ha rebut el missatge

L'autor va consultar per la comunitat d'usuaris de la plataforma B4A quines possibilitats existeixen per crear taules on mostrar la informació llegida pel port CAN i es va optar per incorporar en el projecte la llibreria Table [10] desenvolupada per un dels membres que

forma part de la comunitat B4A i que permet la creació i personalització de taules en funció dels requeriments de l'aplicació.

Així doncs, l'aplicació conté una taula que es inicialitzada en l'activitat d'inicialització que s'executa a l'obrir l'aplicació *Sub Activity_Create* i que a mesura de que es va llegint un missatge CAN, aquest es filtra i s'afegeix en una de les files de la taula, separant la data de lectura del missatge, l'identificador de missatge, la longitud DLC i els 8 bytes de dades.

Aquest procés de lectura i filtració d'informació es fa en la rutina *timer3_tick*, que s'executa cíclicament cada 50ms si es detecta que el port CAN del dispositiu USBtin està obert.

```
'Timer encargado de gestionar la lectura de los mensajes recibidos de CAN
Sub timer3_tick

    msg =canbus1.canReadMessage() 'LO PRINCIPAL, RECUPERAR EL MENSAJE CAN DEL USBTIN


    'Si se recupera un mensaje CAN (en la cola) entonces tratar la información que nos llega
    'y en función de si está habilitado la grabación guardar la información en un fichero
    If msg.Size > 1 Then

        Log(msg)
        Log(msg.Get(0))
        'Readed_MSG.Text = msg
        Dim id, dlc As Int
        Dim D0,D1,D2,D3,D4,D5,D6,D7 As String

        id = msg.Get(1)
        dlc = msg.Get(2)
        bytes = msg.Get(3)
        longitud_bytes = bytes.Length
```

Figura 6-6. Rutina timer3_click executada cíclicament amb l'objectiu de captar el tràfic de dades. Font pròpia

La variable *msg* és una llista que conté els diferents elements llegits del missatge CAN per el dispositiu USBtin. Un cop llegit es duu a terme el procés de filtratge d'aquesta variables i es creen unes variables intermèdies com són: id, dlc, bytes, D0,D1... en les que es copia el valor corresponent a aquests camps de la llista *msg*.

Finalment, un cop tots els elements que conformen el missatge han estat filtrats i copiats a la variable corresponent s'afegeix tota la informació en una fila de la taula a través de la funció *AddRow* de la llibreria Table. A més a més es controla la capacitat d'aquesta taula, ja que es mostren els darrers 18 missatges llegits per el dispositiu.

Això es degut a que l'aplicació té com a objectiu poder mostrar al usuari una petita idea dels darrers missatges que s'han rebut del dispositiu però no tot el trànsit de la xarxa. Per aquest fet es contempla l'opció de la creació de fitxers de dades històriques, tal com s'explica en el següent apartat.

```
'Siempre que llega un mensaje CAN independientemente si esta iniciado el LOG se muestra por la tablet
DateTime.TimeFormat = "dd/MM/yyyy   kk:mm:ss"
Table2.AddRow(Array As String(DateTime.Time(DateTime.now),Bit.ToHexString(id),dlc,D0,D1,D2,D3,D4,D5,D6,D7))
row = row + 1
Log(row)
If row > 18 Then
    Table2.RemoveRow(0)
End If
```

Figura 6-7. Gestió dels missatges CAN llegits i filtració de la informació per mostrar-ho en format taula. Font pròpia

6.2.4. Generació del fitxer històric de dades

Un dels requeriments de l'aplicació CAN-Server és poder emmagatzemar les dades que es reben a través del bus de comunicacions per tal de poder consultar arxius històrics que permetin a l'usuari saber que ha estat transmetent-se per la xarxa on té connectat el dispositiu USBtin.

Val a dir que aquesta funcionalitat afecta tant a nivell local com a nivell remot, ja que l'aplicació guarda els fitxers amb la informació dels missatges CAN en el dispositiu Android on està instal·lada l'aplicació, de forma que estan guardats localment a la memòria interna d'aquest dispositiu. D'altra banda però, amb les funcionalitats remotes que incorpora l'aplicació i que es detallaran en els següents punts, aquest fitxer guardat per l'aplicació podrà ser servit i descarregat per part dels clients que es connectin al *webserver* de l'aplicació, i d'aquesta manera podran descarregar els fitxers històrics que vulguin i fer un anàlisis de la informació sense la necessitat d'estar connectats físicament amb el dispositiu USBtin.

Per tal de crear un històric de dades prèviament és necessari definir quin format ha d'adoptar el fitxer i quines dades ha de guardar. Es creu convenient guardar la mateixa informació que la que es mostra per pantalla alhora que l'aplicació llegeix un missatge CAN, on es captura les dades significatives del missatge rebut, tals com l'ID, la longitud en bytes i la informació dels propis bytes. A més a més, s'emmagatzema l'hora en que s'ha rebut el missatge.

Alhora de definir un format de fitxer s'ha optat per crear un fitxer tipus *CSV*, idoni per representar dades en forma de taula. Així doncs la informació del missatge es separa a traves del separador ";". Per tal de crear un fitxer d'extensió *CSV* i que aquest quan s'obri amb un full de càlcul com pot ser el software Microsoft Office Excel, l'usuari pugui veure la informació emmagatzemada per l'aplicació en forma de taula.

Per tal d'identificar correctament els fitxers d'històric d'errors l'aplicació guarda els fitxers amb la data del moment en que es comença a registrar la informació. A més a més, per tal

de no sobrecarregar la memòria interna de la tableta electrònica es controla a través d'un botó d'usuari si es vol registrar o no els missatges CAN que es llegeixen. D'aquesta manera també l'usuari podrà controlar en quins moments s'ha de guardar la informació.

Així doncs, l'aplicació CAN Server al iniciar l'aplicació, crea un directori anomenat "LOGS" en la memòria interna de la tableta electrònica on s'emmagatzemaran els arxius que es vagin generant. A través de l'esdeveniment que comprova si es prem el botó d'habilitar el registre de dades s'inicialitza un fitxer amb l'encapçalament que fa referència a les dades del missatge CAN que s'emmagatzemen.

```
'Se encarga de definir si se activa o no el LOG de los datos recibidos
Sub ToggleButton1_CheckedChange(Checked As Boolean)

    If ToggleButton1.Checked=False Then
        Logging = False
        TW.Close
    Else If ToggleButton1.Checked=True Then
        TW.Initialize(File.OpenOutput(File.DirRootExternal&"/LOGS", Archivo_Datos, True))
        TW.WriteLine("DATE;TIME;ID;DLC;D0;D1;D2;D3;D4;D5;D6;D7")
        Logging = True
    End If

End Sub
```

Figura 6-8. Esdeveniment que controla el inici del registre dels missatges CAN. Font pròpia

Com es pot comprovar en la Figura 6-8 si es prem el botó s'inicialitza un fitxer a través de la llibreria *TextWriter (TW)*. En la rutina d'inicialització ja s'ha generat prèviament el nom del arxiu que s'emmagatzema a la variable Archivo_Datos i s'ha creat el directori "LOGS" on es guarden aquests arxius.

A més a més, un cop inicialitzat l'arxiu s'activa una variable booleana *Logging* que dins de la rutina on s'analitza si es rep un missatge CAN activarà o no l'escriptura d'una nova línia dins del fitxer amb la informació del missatge en qüestió. Això es gestiona de la forma que es mostra en la Figura 6-9.

```
'Gestión del log de los mensajes recibidos por CAN
'TW.WriteLine(msg)
If Logging = True Then
    DateTime.TimeFormat = "dd/MM/yyyy;kk:mm:ss"
    TW.WriteLine(DateTime.Time(DateTime.now)&";"&Bit.ToHexString(id)&";"&msg.Get(2)&";"&D0&";"&D1&";"&D2&";"&D3&";"&D4&";"&D5&";"
End If
```

Figura 6-9. Rutina amb la que s'afegeix una línia al fitxer d'històric d'errors. Font pròpia

Com es pot comprovar en la Figura 6-9 s'afegeix una línia que equival a un missatge CAN. Entre les diferents dades que es registren es contempla un separador en forma ";", el qual separa la informació i permet generar un fitxer compatible amb l'extensió *CSV*.

Aquest fet permetrà que amb un software de full de càlcul la informació es mostri en forma de taula. D'altra banda, si s'obre un fitxer històric generat per l'aplicació es pot veure com es guarda la informació tal i com s'ha programat en la Figura 6-10.

```
DATE;TIME;ID;DLC;D0;D1;D2;D3;D4;D5;D6;D7
11/06/2016;13:41:32;1;8;11;22;33;44;55;66;77;88
11/06/2016;13:41:32;1;8;11;22;33;44;55;66;77;88
11/06/2016;13:42:00;127;1;AA;;;;;;;
11/06/2016;13:42:09;127;5;AA;22;33;FA;55;;;
11/06/2016;13:42:09;127;5;AA;22;33;FA;55;;;
11/06/2016;13:42:10;127;5;AA;22;33;FA;55;;;
11/06/2016;13:42:10;127;5;AA;22;33;FA;55;;;
11/06/2016;13:42:10;127;5;AA;22;33;FA;55;;;
11/06/2016;13:42:11;127;5;AA;22;33;FA;55;;;
11/06/2016;13:42:11;127;5;AA;22;33;FA;55;;;
11/06/2016;13:42:20;127;2;AA;22;;;;;;
11/06/2016;13:42:21;127;2;AA;22;;;;;;
11/06/2016;13:42:21;127;2;AA;22;;;;;;
11/06/2016;13:42:21;127;2;AA;22;;;;;;
11/06/2016;13:42:21;127;2;AA;22;;;;;;
11/06/2016;13:42:25;127;6;AA;22;33;FA;55;66;;
11/06/2016;13:42:25;127;6;AA;22;33;FA;55;66;;
11/06/2016;13:42:26;127;6;AA;22;33;FA;55;66;;
11/06/2016;13:42:26;127;6;AA;22;33;FA;55;66;;
11/06/2016;13:42:27;127;6;AA;22;33;FA;55;66;;
```

Figura 6-10. Exemple del format d'un fitxer històric de dades. Font pròpia

6.3. Implementació del servidor allotjat a l'aplicació

Com s'ha comentat al llarg de la memòria un dels requeriments de l'aplicació, que de fet és el tret diferencial del projecte respecte als projectes anteriors desenvolupats des del Departament d'Enginyeria Electrònica és la implementació de la funcionalitat *webserver.*

L'aplicació CAN Server té com objectiu permetre que diferents usuaris puguin accedir a la informació que rep l'USBtin de forma remota. Per executar aquesta part el codi font de l'aplicació conté una capa específica que gestiona la creació d'un servidor web emmagatzemat en la pròpia tableta electrònica.

Per generar aquest servidor web s'executa paral·lelament a les funcionalitats locals amb la capa *ServerService,* amb la que s'assigna al servidor web una IP dins d'una xarxa local LAN amb un port específic, per tal de que els clients que estiguin connectats a la mateixa xarxa puguin accedir al servidor web que allotja el dispositiu on s'instal·la l'aplicació CAN Server.

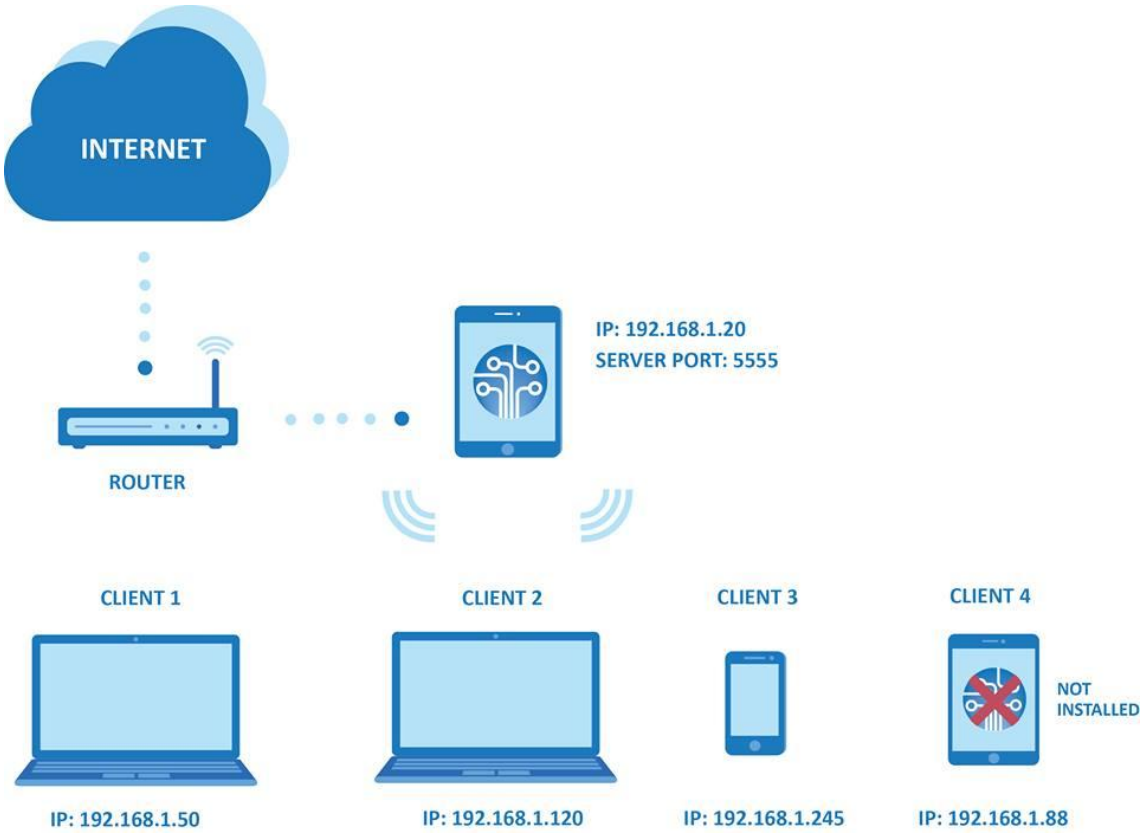


Figura 6-11. Esquema d'una xarxa Wi-Fi amb l'aplicació CAN Server funcionant com a servidor web. Font pròpia

En la Figura 6-11 podem veure un esquema a nivell d'exemple per entendre el funcionament de la part remota de l'aplicació, i es que el dispositiu on s'instal·la l'aplicació es connecta a una xarxa Wi-Fi generada per un punt d'accés o un *router* que assigna a la tableta electrònica una IP. La resta de dispositius "clients" que estan connectats a la mateixa xarxa Wi-Fi i que tenen assignada una IP dins del mateix rang de xarxa es poden connectar al webserver que genera l'aplicació introduint l'IP del dispositiu i el port que utilitza en qualsevol navegador web. Com es pot observar en l'esquema de la Figura 6-11 es poden connectar varis dispositius simultàniament sempre i quan estiguin connectats a la mateixa xarxa.

Per tal d'establir comunicació entre els clients i el servidor s'utilitza el protocol de comunicacions HTTP. Aquest protocol és l'estàndard utilitzat en les transferències d'informació en el *World Wide Web* (Internet). Aquest mètode de comunicació es basa en un sistema de *petició-resposta* entre un client i un servidor. Les peticions que es generen en aquest tipus de protocol estan formades de l'acció que es requereix al servidor seguidament d'una direcció URL.

En el protocol de comunicacions HTTP hi ha diferents mètodes de petició en funció de les accions que es volen dur a terme. En el servidor web implementat s'utilitzen els mètodes GET/POST, per tal d'establir el diàleg i la transferència de dades entre clients i servidor.

Tant el mètode POST com el mètode GET envien una sèrie de dades, per tal de que siguin processades en el servidor. La principal diferència entre aquests dos mètodes es troba en la forma d'enviar aquestes dades al servidor. Tots dos mètodes generen una URL, mentre que el mètode GET envia les dades utilitzant la URL, el mètode POST els envia de forma oculta sense que apareguin a la direcció URL.

Així doncs, basant-se amb aquests mètodes de comunicació, el que es gestiona des de l'aplicació és el seguit de comandes que s'han d'executar en funció del tipus de mètode que li sol·licita el client.

El servidor web que conté l'aplicació organitza els seus continguts en tres pàgines web diferents. Una pàgina web d'accés que demana una identificació al client que es vol connectar (usuari i contrasenya), la pàgina web principal que conté la informació sobre el hardware i permet configurar i gestionar el port CAN de forma remota a la vegada que permet visualitzar els missatges CAN que rep el dispositiu, i una tercera pàgina web que permet als clients descarregar els fitxers que conté un històric dels missatges CAN que rep el dispositiu USBtin.

Val a dir que totes tres pàgines web tenen associat uns fitxers amb extensió *HTML* diferents, els quals s'instal·len amb l'aplicació i es guarden en la memòria del dispositiu. Així doncs, quan es precís iniciar la part de servidor web, es carrega el fitxer *HTML* de la pàgina d'accés. A continuació si el client accedeix a la pàgina web principal es carrega per complet el fitxer *HTML* associat, en aquest punt però, un cop carregada la pàgina web principal es duu a terme una actualització dinàmica de les parts de la pàgina en les que és necessari refrescar la informació de les variables que mostren es mostren. És el cas de l'estat del port CAN, la comunicació amb el dispositiu USBtin, els missatges que s'estan rebent al hardware...

L'objectiu final en la part remota del projecte es basa en intentar calcar les funcionalitats de la part local de l'aplicació. És per això que la part de navegació remota ha d'incloure la gestió dels següents punts:

- ✓ Permetre als clients que es connectin configurar la connexió amb el port CAN
- ✓ Mostrar l'estat de la connexió del hardware USBtin.
- ✓ Lectura en temps real dels missatges CAN rebuts
- ✓ Possibilitat de descàrrega de fitxers històrics amb la informació dels missatges CAN.

6.3.1. Gestió dels mètodes requerits des de l'aplicació

Per tal de dur a terme les funcionalitats presentades en l'anterior punt, com s'ha comentat anteriorment, s'utilitza el protocol de comunicacions HTTP per tal de transferir la informació i les comandes entre els clients i el servidor web. Per gestionar tot aquests processos d'intercanvi d'informació dins del servei *ServerService* hi ha la subrutina *Server_HandleRequest* que gestiona i filtra les diferents peticions (*Requests)* que s'intercanvien entre la pàgina web i els usuaris, que són de tipus GET i POST.

Els mètodes utilitzats en l'aplicació per executar peticions d'intercanvi d'informació entre client i servidor es basen en que quan es produeix la petició es genera una direcció URL que el servidor rep i pot filtrar i d'aquesta manera saber quin mètode i quina acció ha de realitzar per satisfer la petició generada per el client.

Cal veure que existeix en aquest punt una combinació entre el codi programat en llenguatge HTML que mostra la pàgina web i el codi implementat en l'aplicació en el llenguatge de la plataforma B4A.

6.3.1.1. Gestió de la càrrega de la pàgina web principal

Un cop inicialitzat el servei que gestiona la part del servidor web allotjat en l'aplicació d'entre els diferents *Requests* que rep l'aplicació, hi ha la necessitat en un primer instant de carregar la pàgina web d'inici quan un client es connecta a l'adreça IP del servidor web. Per tal de no carregar directament la part de la web que mostra tota la informació referent a la xarxa de comunicacions CAN, el dispositiu USB i els diferents fitxers històrics, s'implementa una pàgina web d'accés protegida amb usuari i contrasenya amb nivell bàsic de seguretat.

De fet es tracta d'un fitxer HTML que es carrega des de l'aplicació i s'envia al navegador web d'on s'ha connectat un client. El fitxer de codi HTML forma part d'un dels arxius que s'afegeix al dispositiu amb la instal·lació de l'aplicació i quan es necessari, es mostra al client que ho requereix.

Al tractar-se d'una pàgina d'accés amb un nivell de seguretat mínim, aquest es gestiona des del codi HTML i no des de l'aplicació. La pàgina consta de dos camps en el que s'han d'introduir el text d'usuari i el valor de la contrasenya. El codi HTML contempla un petit *script* que s'encarrega de comparar el text introduït per l'usuari amb uns valors prefixats. Si la comparació resulta satisfactòria, *l'script* s'encarrega d'executar l'ordre de carregar el fitxer que allotja la pàgina principal amb la informació rellevant de la xarxa de comunicacions CAN.

D'altra banda, si la comparació no és correcta s'avisa al usuari que no ha introduït correctament el usuari ni la contrasenya correctes.

```
<script type="text/javascript">
function go(){

if (document.form.password.value=='1234' && document.form.login.value=='user'){
        document.form.submit();
    }
    else{
        alert("Porfavor ingrese, nombre de usuario y contraseña correctos.");
    }
}
</script>
```

Figura 6-12. Codi HTML que gestiona l'accés a la pàgina web principal del servidor. Font pròpia

6.3.1.2. Gestió de l'estat del port CAN mitjançant mètodes POST

S'inclou la gestió de dos mètodes POST que es generen des de la pàgina web quan l'usuari prem un botó. En el servidor web hi ha dos botons: un destinat a obrir el port CAN i seleccionar una velocitat de transmissió concreta i un altre per tancar el port CAN. La rutina filtra quin dels dos botons ha generat un mètode POST filtrant l'acció corresponent al botó concret tal i com es pot apreciar en la Figura 6-13.

```
Sub Server_HandleRequest (Request As ServletRequest, Response As ServletResponse)
    Try
        Log("Client: " & Request.RemoteAddress & ", Request: " & Request.RequestURI)
        If Request.Method = "POST" Then 'User pressed on one of the buttons
            action = Request.RequestURI
            Log(action)
            Select True
                Case action.StartsWith("/sendbutton1")
                    Dim baud As String = Request.GetParameter("baudOption")
                    Log(baud)
                    CanOPEN_remotely(baud)
                Case action.StartsWith("/sendbutton2")
                    CanCLOSE_remotely
                'Case action.StartsWith("/sendbaudrate")
                '    Dim baud As String = Request.GetParameter("baudOption")
                '    Log(baud)
            End Select
            Response.SendRedirect("/mainpage3")
```

Figura 6-13. Gestió dels mètodes POST generats des de la pàgina web. Font pròpia

Per entendre correctament el funcionament d'aquesta part de codi s'ha de centrar-se en el fet de que quan l'usuari prem un dels botons en la web internament es genera el mètode POST i en funció de quin s'ha pitjat s'envia l'acció /sendbutton1 o /sendbutton2. Aquest *string* es rebut per l'aplicació que està pendent dels *Requests* que demanden els clients i

filtra les accions a realitzar en funció del *Request* que es rep.

En el cas del botó encarregat d'inicialitzar el port CAN des de codi HTML es genera l'acció corresponent (*sendbutton1*) i a més a més s'envia un paràmetre que recull la velocitat de transmissió que s'ha escollit per obrir el port CAN utilitzant l'opció que permet el servei *Request* d'adquirir un paràmetre dins d'un mètode POST. Aquest paràmetre es definit des del codi HTML amb el nom de "baudOption".

```
<form method='POST' enctype="multipart/form-data" action="sendbutton1">
    Baud:
    <select name="baudOption">
        <option value="0" selected >10 kBaud</option>
        <option value="1">20 kBaud</option>
        <option value="2">50 kBaud</option>
        <option value="3">100 kBaud</option>
        <option value="4">125 kBaud</option>
        <option value="5">250 kBaud</option>
        <option value="6">500 kBaud</option>
        <option value="7">800 kBaud</option>
        <option value="8">1 MBaud</option>
    </select><br>
    <input type="submit" id="btn_open" name="btn_open" value="CAN OPEN">
</form>

<form method='POST' enctype="multipart/form-data" action="sendbutton2">
    <input type="submit" id="btn_close" name="btn_close" value="CAN CLOSE">
</form>
<hr/>
```

Figura 6-14. Gestió dels mètodes POST de la pàgina web mitjançant codi HTML. Font pròpia

En la **Figura 6-14** es pot observar el codi HTML que gestiona els botons de la pàgina web. Com es pot apreciar, es defineix per cada mètode POST l'acció corresponent a més d'enviar-se el paràmetre corresponent al *baudOption* que s'escull d'una llista desplegable que conté les diferents opcions.

D'aquesta forma quan es vol obrir el port CAN l'aplicació emmagatzema l'acció generada per el mètode POST dins de la variable *action* i filtrant l'acció que es requereix, executa la crida a la subrutina CanOPEN_remotely passant com a paràmetre la velocitat de transmissió que al seu torn té la funció d'executar les comandes per obrir el port CAN.

```
Sub CanOPEN_remotely(baud As Object)

    CallSub2(Main, "OpenCAN", baud)

End Sub
```

Figura 6-15. Crida des del servei ServerService a les rutines de gestió del port CAN. Font pròpia

Amb la funció *CallSub2* el que es realitza és la crida d'una rutina que ha estat implementada en un altre mòdul de l'aplicació. Com que de forma remota ja s'inicialitza el port CAN a una velocitat de transmissió concreta en el mòdul *Main*, el que es fa en el servei *ServerService* un cop es rep la informació corresponent a que un client ha premut el botó d'obrir el port CAN, és executar aquesta subrutina *CanOPEN_remotely* que al seu torn executa la funció corresponent del *Main OpenCAN*.

Per dur a terme l'acció contrària, és a dir, la de tancar el port CAN del dispositiu, el funcionament és molt similar. El canvi radica en la petició que genera el client a través de la pàgina web. La petició creada en aquest cas té associada la direcció URL "/sendbutton2", que quan es rebuda al servidor, executa la subrutina encarregada de tancar el port.

6.3.1.3. Gestió de la visualització de diferents variables mitjançant mètodes GET

En la pàgina web principal també es mostra informació referent a l'estat del dispositiu, l'estat del port CAN, la velocitat de transmissió seleccionada i com no, la informació procedent dels missatges CAN que rep el dispositiu USBtin.

Es tracta d'una sèrie de variables contingudes en l'aplicació, les quals s'han de mostrar als usuaris per tal de que es facin una idea de quin és l'estat del sistema amb el que estan treballant. És per això que l'estat d'aquestes variables apareix en la pàgina web principal i el servidor ha de treballar per tal de que s'actualitzi de forma automàtica el seu valor.

Per tal de dur a terme aquest fet, s'utilitza un *script* des del codi font de la pàgina principal *mainpage3.html*, que s'encarrega de generar peticions de forma automàtica cada cert temps (en el cas actual aquest interval de temps d'actualització és fixa a 1 segon).

Aquest *script* s'utilitza per generar tres peticions concretes associades a una direcció URL determinada per cada una d'elles amb la fi d'actualitzar les tres cadenes de caràcters següents *(strings).*

1. **USBStatus**

Es tracta d'una cadena de caràcters que conté l'estat de la comunicació USB entre la tableta electrònica i el hardware USBtin. Pot prendre dos valors diferents en funció de si la comunicació USB està establerta o no.

Quan l'*script* d'actualització de variables genera la petició, el servidor web rep la direcció URL "/usbstatus" i des de la rutina *Server_HandleRequest* s'executa l'acció d'actualitzar l'estat de la variable USBStatus.

2. CANStatus

Es tracta d'una cadena de caràcters que conté tant l'estat del port CAN del hardware com la velocitat de transmissió a la que treballa el port CAN. Quan l'*script* d'actualització de variables genera la petició, el servidor web rep la direcció URL "/canstatus" i des de la rutina *Server_HandleRequest* s'executa l'acció corresponent per tal d'actualitzar l'estat del port i la velocitat de transmissió.

3. Lectura de missatges CAN rebuts

En la pàgina web es mostra, al igual que en l'aplicació a nivell local, els missatges CAN que es reben pel port CAN del dispositiu USBtin. Es vol mostrar aquesta informació en forma de taula, per la qual cosa quan el servidor rep la petició d'actualització corresponent als missatges amb direcció URL "/messages", s'executa una rutina la qual genera una taula i cada cop que es rep un nou missatge s'afegeix en una fila de dita taula.

El funcionament d'aquesta part de l'aplicació és senzill. Quan es rep un missatge CAN, en la rutina principal explicada en l'apartat 6.2.3 es crea una llista en la que a cada cop que es rep un missatge s'hi afegeix la informació corresponent del ID del missatge, la longitud en bytes DLC, l'hora i data en la que s'ha rebut i les dades pròpies dels bytes que s'envien.

Així doncs, es té la informació en la llista *Messages* i a cada volta que es rep la petició d'actualització es tracta i filtra aquesta llista per mostrar-la en una taula amb els diferents camps dels missatges rebuts.

Com en el cas de la visualització a nivell local es limita a un nombre màxim els missatges CAN que es poden mostrar de forma *online*, per evitar problemes de sobrecarrega d'informació. Per dur a terme aquest fet es gestiona una cua *FIFO* en la que es mostren els darrers 10 missatges que ha rebut el dispositiu. Per tal de consultar més informació sobre els missatges CAN rebuts es pot consultar als fitxers històrics en els que s'emmagatzemen tots els missatges quan la funció de registrar està activada.

Cal comentar que per tal de que en el servidor web els missatges apareguin correctament i en el format esperat (ja sigui en negreta, en el format taula amb els diferents camps dels missatges CAN), les cadenes de caràcters que contenen les variables a visualitzar estan adaptades al llenguatge HTML.

És a dir, des de la plataforma B4A es prepara les variables a través de programació afegint els caràcters adients per tal de que adoptin un format que comprengui el codi HTML de la pàgina web. En la figura Figura 6-16 es mostra un exemple del contingut que adopta la llista dels missatges CAN per tal de crear una taula en la pàgina web.

```
sb.Append("<TABLE BORDER>")
sb.Append(" <col width="&"200"&"> <col width="&"75"&"> <col width="&"50"&"> <col width="&"50"&"> <col width='
sb.Append("<tr> <th>"&"Date & Time"&"</th> <th>"&"ID"&"</th> <th>"&"DLC"&"</th> <th>"&"D0"&"</th> <th>"&"D1"&
For Each s As String In Messages
    sb.Append(s)
Next
sb.Append("</TABLE>")
```

Figura 6-16. Part de l'aplicació B4A on s'afegeix llenguatge HTML en una variable de l'aplicació

Per tal de crear una taula i que aquesta es pugui mostrar en una pàgina web ha de tenir el format específic per a que el llenguatge HTML ho entengui i ho mostri com a tal. És per això que s'afegeixen caràcters com *"<col width="*> per definir l'amplitud d'una columna, el caràcter *<th>&ID</th>* que corresponent al text que incorporarà la cel·la corresponent de la taula, entre d'altres.

És a dir, es combina codi HTML des del propi codi de l'aplicació en llenguatge B4A, ja que quan s'envia la cadena de caràcters aquesta substitueix una variable concreta dins del codi font HTML i llavors la pàgina web executa el contingut d'aquesta variable com a codi propi i crea la visualització corresponent.

6.3.1.4. Gestió de l'accés i la descàrrega de fitxers d'històrics de dades

Un dels altres requeriments de l'aplicació a nivell remot es basa en l'obtenció dels fitxers que contenen tota la informació referent als missatges CAN que es reben a través del dispositiu USBtin.

En l'apartat 6.2.4 es detalla com es generen aquests fitxers d'històrics de dades en funció de si l'usuari de l'aplicació decideix que s'activi la gravació de missatges o no. Aquests

fitxers queden guardats en la memòria de la tableta electrònica on està instal·lada l'aplicació, però en aquest punt es presenta la forma en la que des de forma remota els clients que es connectin al servidor web de l'aplicació poden visualitzar i descarregar aquests fitxers emmagatzemats al dispositiu.

Per dur a terme aquestes accions, el servidor web emmagatzema una pàgina específica que mostra el contingut de la carpeta on es guarden els fitxers. Per accedir a aquesta pàgina, des de la pàgina principal hi ha un enllaç que crea una petició al servidor amb la direcció URL "/list", per tal de que mostri la pàgina en qüestió i executi les rutines corresponents.

Quan el servidor rep la petició executa la rutina *HandleList* que és l'encarregada de filtrar i mostrar el contingut de la carpeta /LOGS, que està emmagatzemada en la memòria de la tableta i és on es generen i guarden el conjunt de fitxers que contenen la informació referent als missatges CAN rebuts.

Alhora de mostrar-los per la pàgina web utilitzant un procediment similar al de combinar codi HTML dins de l'aplicació B4A, s'afegeix l'opció de veure el fitxer creant un enllaç al costat de cada fitxer per tal de que si l'usuari el prem, es mostri el contingut del fitxer.

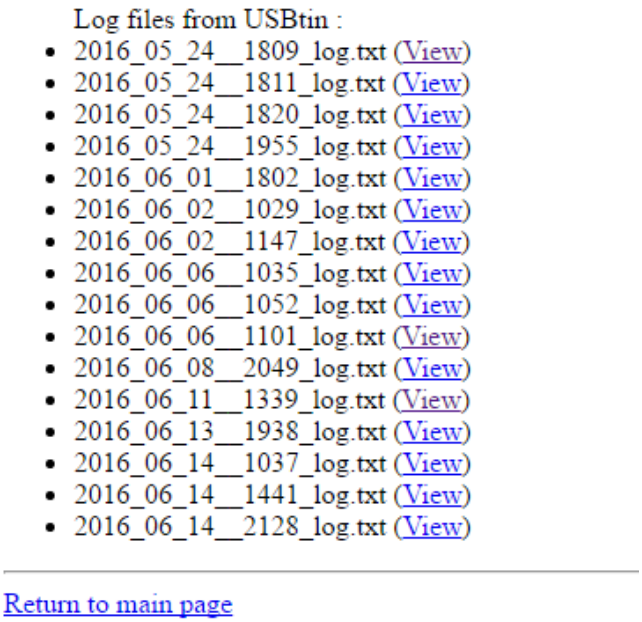


Figura 6-17. Aspecte de la pàgina web que mostra els diferents fitxers emmagatzemats en el dispositiu. Font pròpia

Un cop s'obre un dels fitxers, es pot veure el seu contingut a través del navegador web. Aquest contingut es pot descarregar de la mateixa forma que es faria amb una imatge o un fitxer d'Internet a través de la opció de descàrrega o guardar que ofereix el navegador web que s'estigui utilitzant.

Es tracta d'un fitxer de text pla, que es pot guardar amb l'extensió *CSV* per tal de que després pugui ser obert amb un editor de text com pot ser el conegut Microsoft Office Excel, ja que tots els camps estan separats per un limitador ";".

7. Test i validació

En aquest apartat de la memòria es descriuen els procediments i les proves realitzades per tal de validar el correcte funcionament de les funcionalitats implementades en l'aplicació Android desenvolupada.

Tot i que durant la fase de desenvolupament, tant de la part local com de la part de desenvolupament del servidor web, es van realitzar de forma paral·lela proves per tal de comprovar el funcionament de l'aplicació, és al final de la fase de desenvolupament de l'aplicació quan es realitza un test complet del projecte i en aquest punt de la memòria es descriuen els procediments i els resultats obtinguts.

7.1. Configuració utilitzada

Alhora de realitzar les proves de validació de l'aplicació ha estat necessari adoptar una configuració que permetés simular el funcionament d'una xarxa de comunicacions CAN en la qual s'estigués transmetent missatges per la xarxa. Degut a la impossibilitat de disposar d'una maquinà real que incorporés una xarxa de comunicacions amb protocol CAN es va realitzar una simulació utilitzant un dels productes citats en l'estat de l'art, concretament, el hardware PCAN-USB amb el seu software associat PCAN-View [4].

El hardware PCAN-USB es connecta al hardware utilitzat en el projecte USBtin, tal com es mostra a la Figura 7-1 . D'aquesta manera es simula una xarxa CAN, ja que gràcies a l'eina software PCAN-View es permet enviar una sèrie de missatges CAN que circulen a través del PCAN-USB fins a la xarxa CAN, i alhora la targeta USBtin capta aquests missatges.

D'altra banda, per tal de fer les proves des de l'aplicació CAN Server, aquesta s'instal·la en una tableta proporcionada des del Departament d'Enginyeria Electrònica de l'ETSEIB, que disposa d'una tableta *Wolder Chicago*. Aquesta tableta disposa d'un port USB en el que es connecta la targeta USBtin.

Finalment, per fer les proves de la part del servidor web, es disposa d'un PC portàtil i una tableta en la que no està instal·lada l'aplicació però que té accés a connexió Wi-Fi i un navegador web amb el que poder accedir al servidor web i provar d'aquesta forma el funcionament.

En la Figura 7-1 es pot veure una fotografia del muntatge dut a terme en el període de proves i validació de l'aplicació amb els components citats en aquest apartat.

1. Ordinador portàtil
2. Tableta Samsung sense l'aplicació CAN Server instal·lada
3. Hardware convertidor PCAN-USB
4. Targeta USBtin
5. Tableta Wolder Chicago amb l'aplicació CAN Server instal·lada

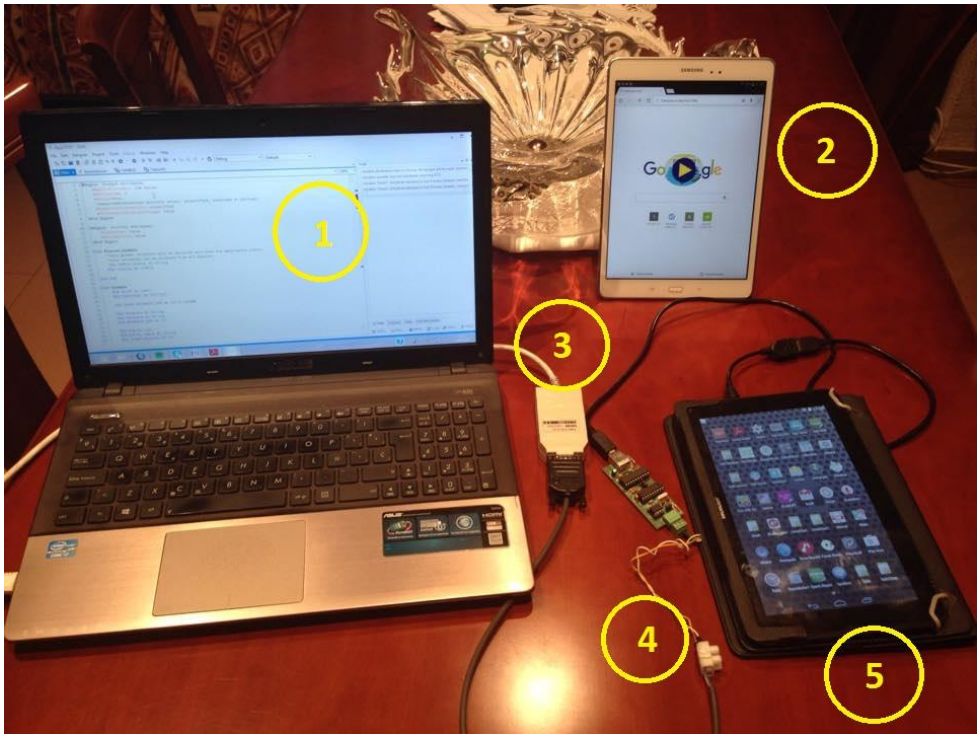


Figura 7-1. Muntatge utilitzat en el procediment de test i validació de l'aplicació. Font pròpia

Per tal de dur a terme les proves es transfereix el projecte creat a través de la plataforma B4A a la tableta *Wolder*. Per fer aquest procediment s'instal·la l'aplicació B4A-Bridge (Figura 7-2) a la tableta i s'enllaça l'ordinador i la tableta a través de la direcció IP que té assignada l'aplicació.

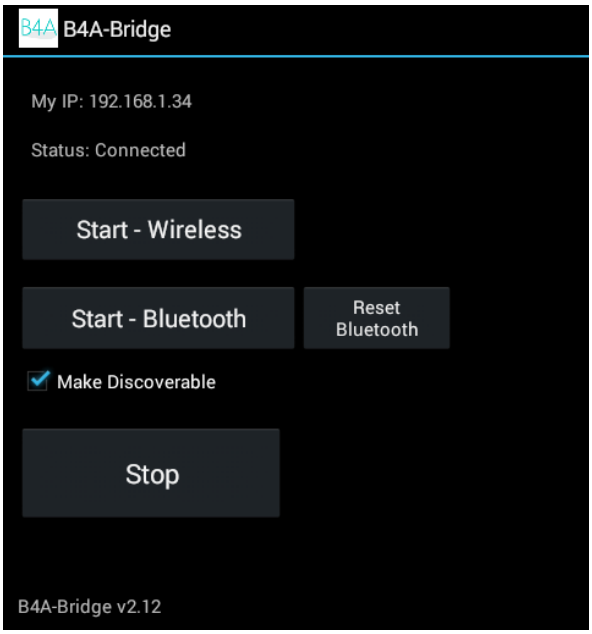


Figura 7-2. Captura de pantalla de l'aplicació B4A-Bridge per dur a terme la instal·lació de l'aplicació. Font pròpia

Un cop realitzat el procediment d'instal·lació de l'aplicació mitjançant l'enllaç realitzat amb l'aplicació B4A-Bridge, en el menú principal apareix la icona de l'aplicació CAN Server (Figura 7-3).

En aquest moment qualsevol usuari pot començar a executar-la. Val a dir però que durant les proves de validació s'instal·la l'aplicació en mode *Debug* per tal que des de la plataforma B4A es pugui testejar i veure el valor que adopten les variables, quins mètodes s'estan requerint al servidor web, forçar l'execució pas a pas mitjançant l'introducció de punts de ruptura (*breakpoints)* entre d'altres avantatges de cara al desenvolupament.

Val a dir però, que com a contrapartida, durant aquesta fase de validació l'usuari que vulgui executar l'aplicació en mode *debug* ha d'estar necessàriament connectat mitjançant l'aplicació B4A-Bridge amb l'ordinador on hi ha el projecte B4A de l'aplicació CAN Server.

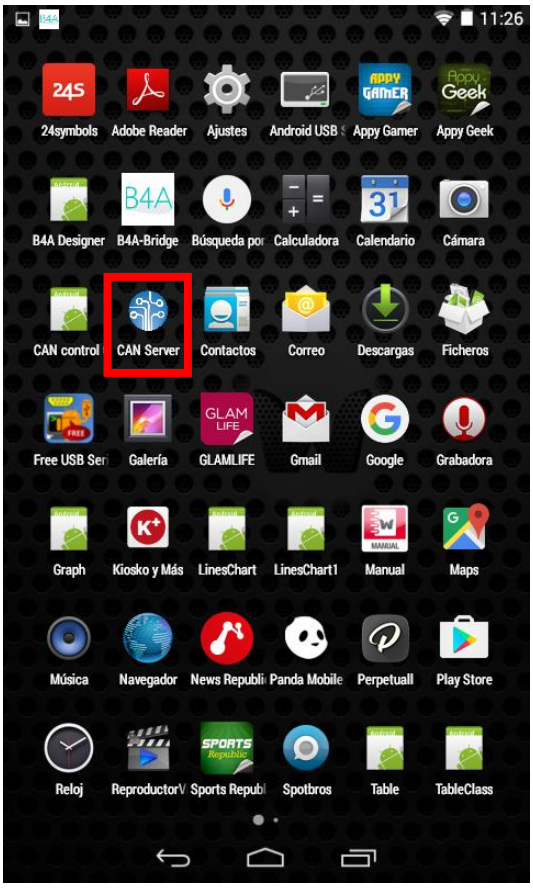


Figura 7-3. Menú principal del dispositiu on s'instal·la l'aplicació CAN Server. Font pròpia

7.2. Proves de funcionament a nivell local

En una primera fase de la validació, es realitza una prova de funcionament utilitzant la interfície gràfica creada en l'aplicació per tal de connectar-se a la targeta USBtin,obrint el port CAN a diferents velocitats de transmissió i observantquins resultats s'obtenen.

Per fer aquesta prova, de forma paral·lela es genera un tràfic de diversos missatges CAN des de l'eina PCAN-View que s'enviaran cíclicament a través de la xarxa CAN. En la Taula 7-1 es defineix quins missatges CAN s'enviaran per tal d'executar la prova de funcionament local.

ID	DLC	D1	D2	D3	D4	D5	D6	D7	D8	Tiempo de ciclo (s)
0x181	4	11	22	33	44					1.0
0x281	5	1A	2B	3C	4C	5C				2.5
0x41A	6	10	20	30	40	50	60			4.0
0x51A	7	5	10	15	20	25	30	FF		6.5
0x6FF	8	11	22	33	44	55	66	77	88	8.0

Taula 7-1. Missatges enviats a la xarxa CAN per dur a terme la simulació de tràfic. Font pròpia

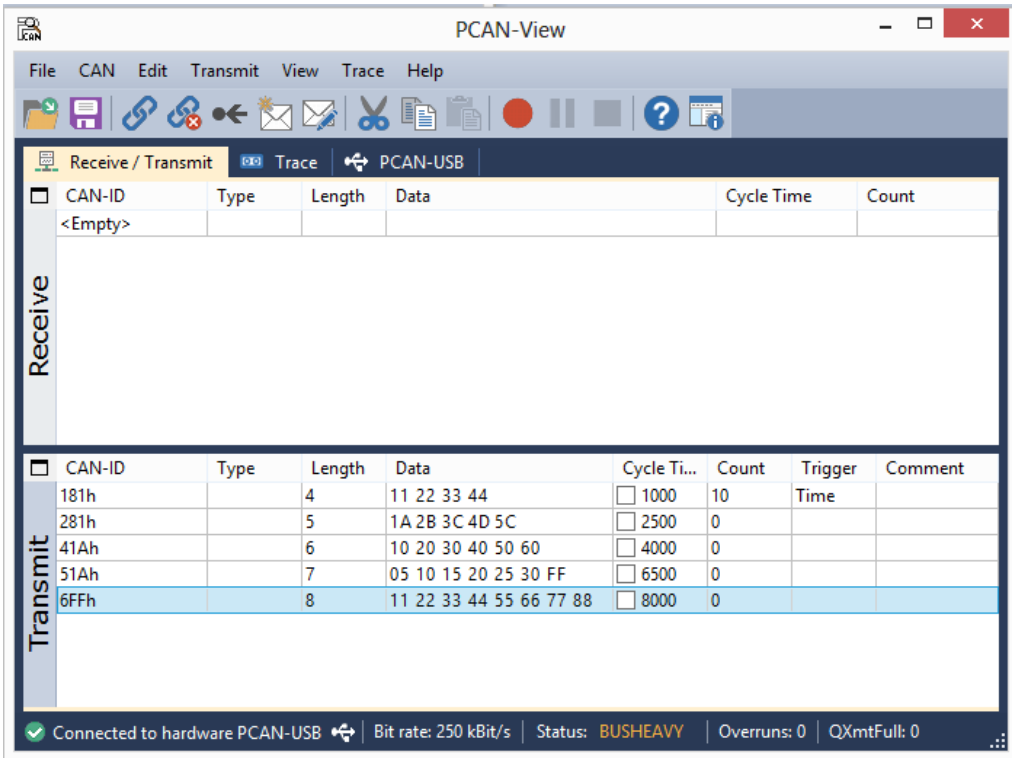


Figura 7-4. Configuració dels missatges de simulació amb el software PCAN-View. Font pròpia

Al iniciar l'aplicació per primer cop, sempre amb el hardware connectat al port USB, es demana permisos per tal de poder utilitzar aquest port i establir d'aquesta forma la comunicació amb el hardware USBtin. Un cop habilitats el permís d'utilització del port USB l'aplicació inicia el procés de connexió i queda a l'espera de que l'usuari interactuï amb ell mitjançant la interfície gràfica creada.

En aquesta prova de funcionament el que es realitza és l'obertura del port CAN a una velocitat de 250Kb/s, ja que es la que es defineix a l'altra banda de xarxa per crear la simulació de missatges a través del dispositiu PCAN-USB. I un cop obert el port, es comencen a rebre els missatges generats per la simulació.

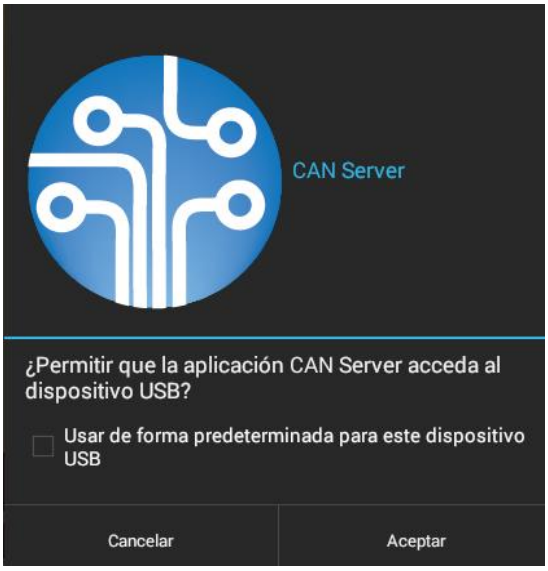


Figura 7-5. Pop-up per demanar permís d'utilització del port USB de la tableta. Font pròpia

El resultat de la prova es pot veure en les en la Figura 7-6, Figura 7-7, Figura 7-8 i Figura 7-9. on s'adjunten les captures de pantalla de la interfície gràfica de l'aplicació amb els diferents estats en els que s'ha passat durant el test del funcionament a nivell local.

Un cop configurat el port CAN a una velocitat de 250Kb/s es comença a rebre i mostrar els missatges CAN que està transmetent el hardware PCAN-USB. Aquests són rebuts i visualitzats correctament per la pantalla de la tableta. A més a més, es comprova que una vegada tancat el port CAN es deixa de rebre la informació al instant.

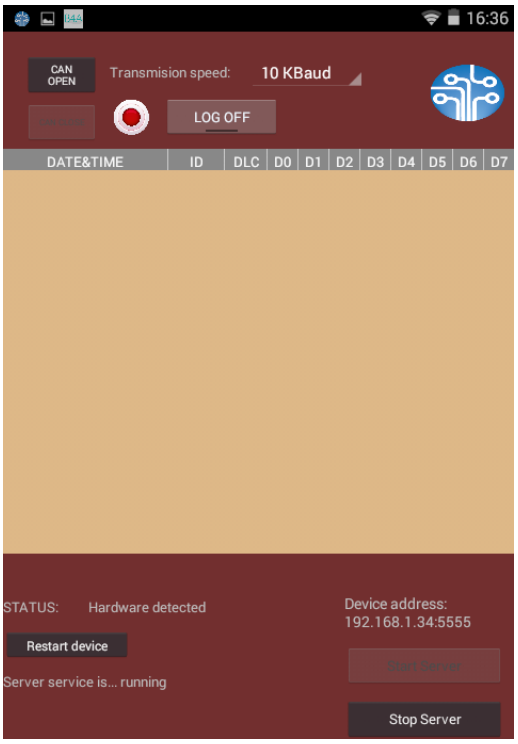


Figura 7-6. Lay-out de l'aplicació. Font pròpia

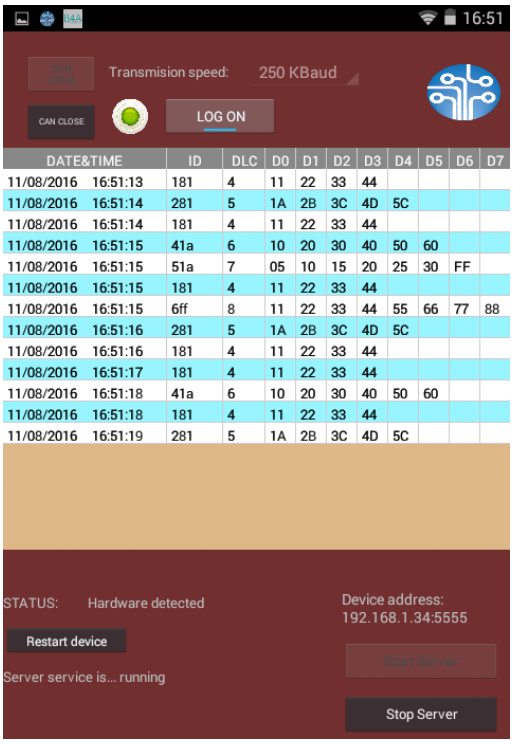


Figura 7-7.Moment en el que s'obra el port CAN. Font pròpia

Figura 7-8. Recepció de missatges CAN durant la prova. Font pròpia

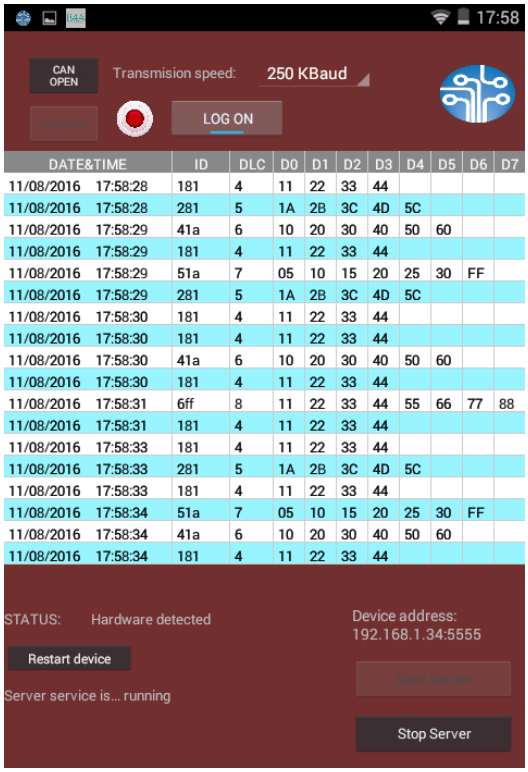


Figura 7-9.Tancament del port CAN al finalitzar la prova. Font pròpia

Un altre dels punts testejats mitjançant la realització d'aquesta prova és la generació de fitxers històrics. Es pot comprovar que dins de la memòria interna del dispositiu es crea la carpeta "/LOGS", on es guarden els fitxers amb nom corresponent a la data i hora de la seva creació quan l'usuari selecciona l'opció de registrar les dades.

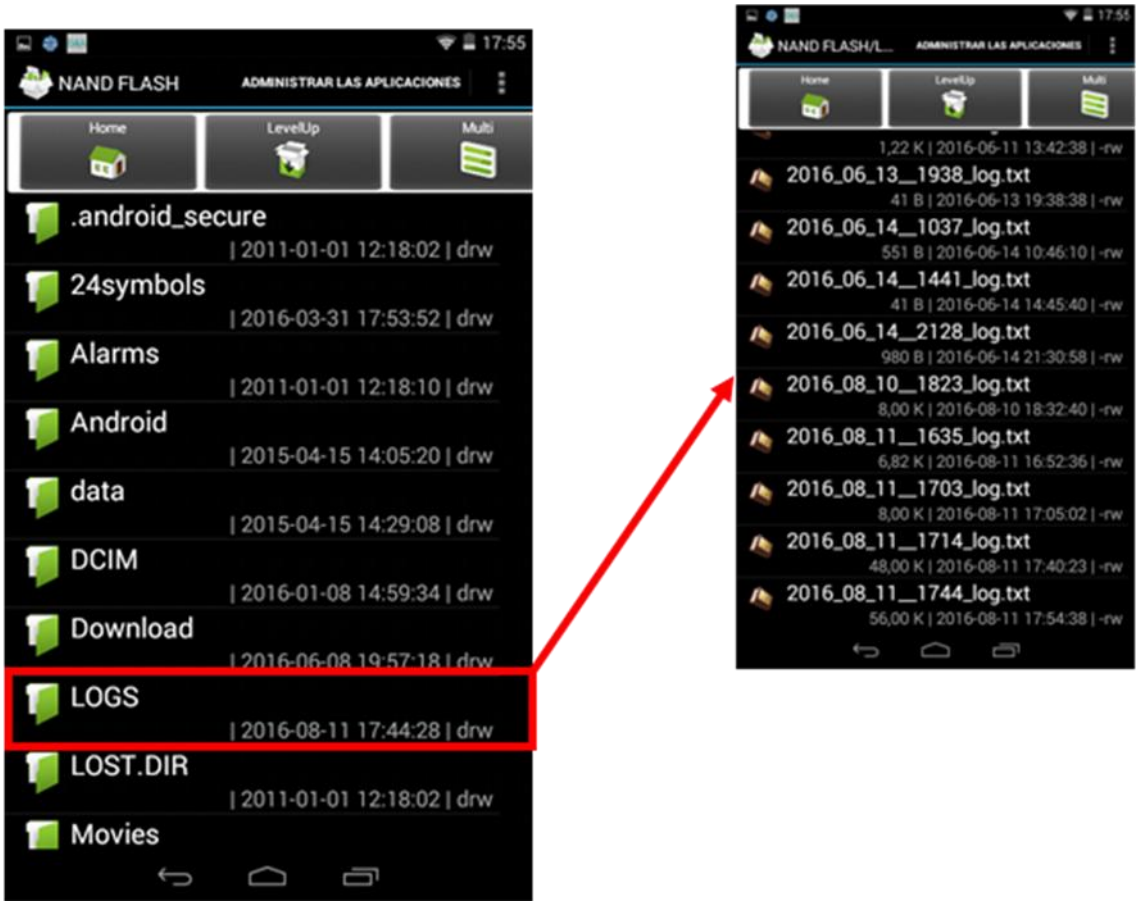


Figura 7-10. Captura de pantalla de la memòria interna del dispositiu on es guarden els fitxers d'històrics de dades. Font pròpia

El resultat de la prova de funcionament a nivell local resulta satisfactòria, ja que s'aconsegueix establir comunicació amb la targeta USBtin, manipular i controlar l'estat del port CAN i la seva velocitat de transmissió, i es permet al usuari veure els tràfic de dades de la xarxa de comunicacions alhora que se li permet guardar tota aquesta informació en un històric de dades.

7.3. Prova de funcionament del servidor web allotjat a l'aplicació

Un cop executada la prova de funcionament a nivell local i essent aquesta satisfactòria, cal passar a validar el funcionament del servidor web que allotja l'aplicació CAN Server. Per dur a terme aquesta prova es realitza un procediment similar que en l'anterior apartat, simulant el tràfic de missatges CAN a través del hardware PCAN-USB, que s'encarrega d'enviar els missatges definits en la Taula 7-1.

Així doncs, un cop es té l'aplicació en funcionament, es pot observar en la part inferior dreta de la interfície de la tableta on està instal·lada la informació referent per tal de que els clients es puguin connectar al servidor web.

Per dur a terme la validació del servidor web es fa mitjançant la connexió simultània de dos clients diferents. Un dels clients es connectarà a través del ordinador portàtil que es pot veure en la imatge de la configuració utilitzada, i l'altre client es connectarà a través d'una tableta on no està instal·lada l'aplicació CAN Server, tal i com s'indica també en la Figura 7-1 on apareix la configuració utilitzada per el procediment de validació.

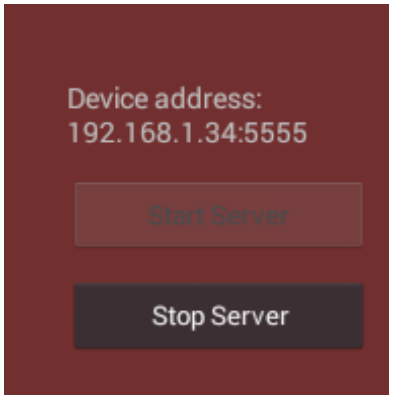


Figura 7-11. Informació per l'accés al servidor web des d'un navegador. Font pròpia

Quan l'usuari que es vol connectar al servidor web introdueix l'adreça IP amb el port d'accés, tal i com es mostra en la Figura 7-11, el resultat és que accedeix a la pàgina de *login* on ha d'introduir les credencials correctes per accedir a la pantalla principal del servidor web. Si les credencials introduïdes no són les correctes, el servidor web mostra un avís de que s'ha produït un intent d'accés erroni.

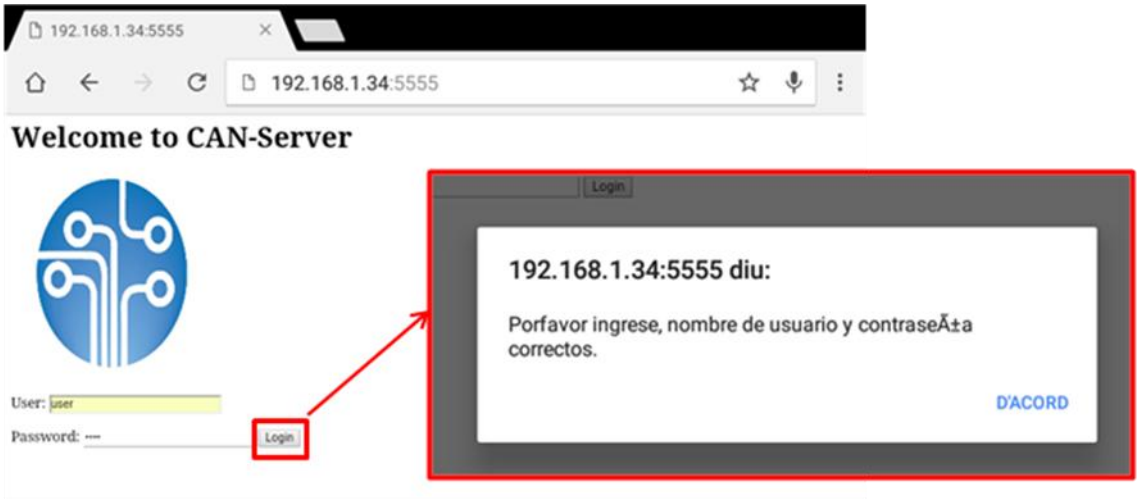


Figura 7-12. Pàgina web d'accés al servidor amb resultat erroni. Font pròpia

Un cop es produeix l'accés al servidor de forma correcta es mostra al usuari el contingut del servidor web. En la prova de validació es pot comprovar en la Figura 7-13 com s'està connectat al maquinari USBtin i interactuar amb ell obrint el port CAN i tancant-lo. A més a més, s'actualitza la informació cada segon i es comprova que s'estan rebent els missatges corresponents a la simulació desenvolupada.

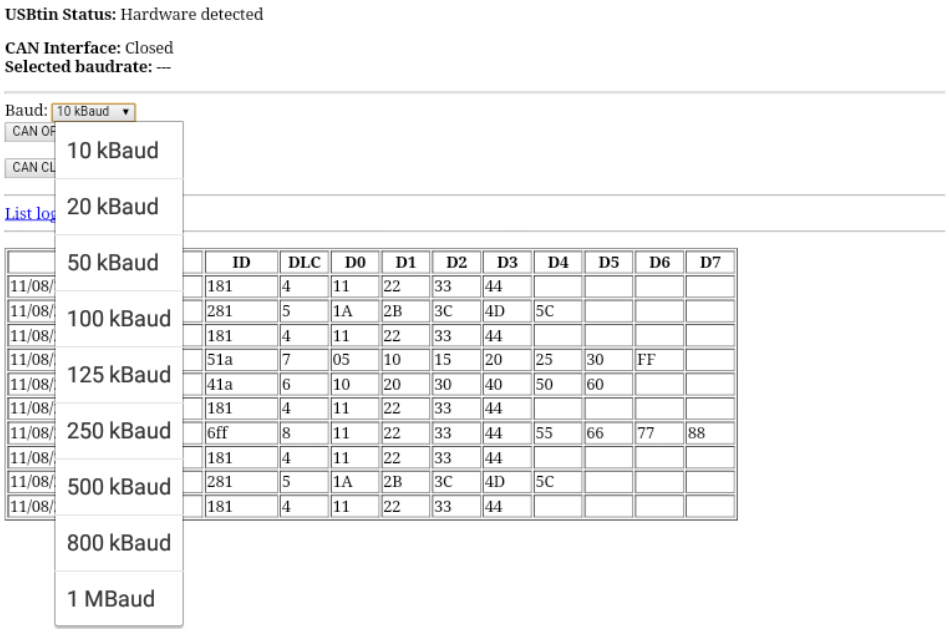
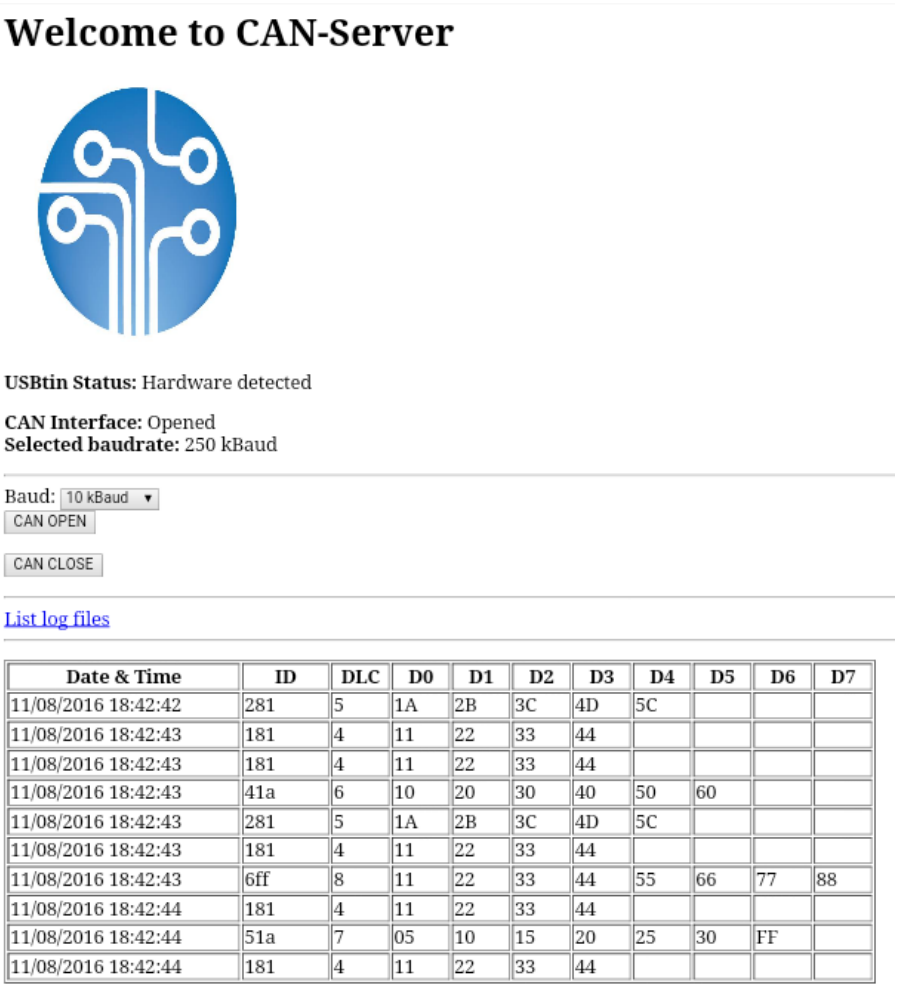


Figura 7-13. Captura de pantalla del moment que l'usuari escull la velocitat de transmissió de forma remota. Font pròpia



Welcome to CAN-Server

USBtin Status: Hardware detected

CAN Interface: Opened
Selected baudrate: 250 kBaud

Baud: 10 kBaud
CAN OPEN

CAN CLOSE

List log files

Date & Time	ID	DLC	D0	D1	D2	D3	D4	D5	D6	D7
11/08/2016 18:42:42	281	5	1A	2B	3C	4D	5C			
11/08/2016 18:42:43	181	4	11	22	33	44				
11/08/2016 18:42:43	181	4	11	22	33	44				
11/08/2016 18:42:43	41a	6	10	20	30	40	50	60		
11/08/2016 18:42:43	281	5	1A	2B	3C	4D	5C			
11/08/2016 18:42:43	181	4	11	22	33	44				
11/08/2016 18:42:43	6ff	8	11	22	33	44	55	66	77	88
11/08/2016 18:42:44	181	4	11	22	33	44				
11/08/2016 18:42:44	51a	7	05	10	15	20	25	30	FF	
11/08/2016 18:42:44	181	4	11	22	33	44				

Figura 7-14. Captura de pantalla de la simulació del funcionament amb lectura del tràfic de dades de la xarxa CAN. Font pròpia

L'altre punt a validar en la part del servidor web fa referència a l'adquisició dels fitxers de dades que genera l'aplicació i que estan emmagatzemats en la memòria interna del dispositiu on està executant-se el CAN Server. Per dur a terme aquesta validació s'accedeix a la pàgina web que mostra el contingut de la carpeta "/LOGS" i es descarrega al portàtil un dels fitxers que ha generat l'aplicació, tal i com es pot veure a la Figura 7-15.

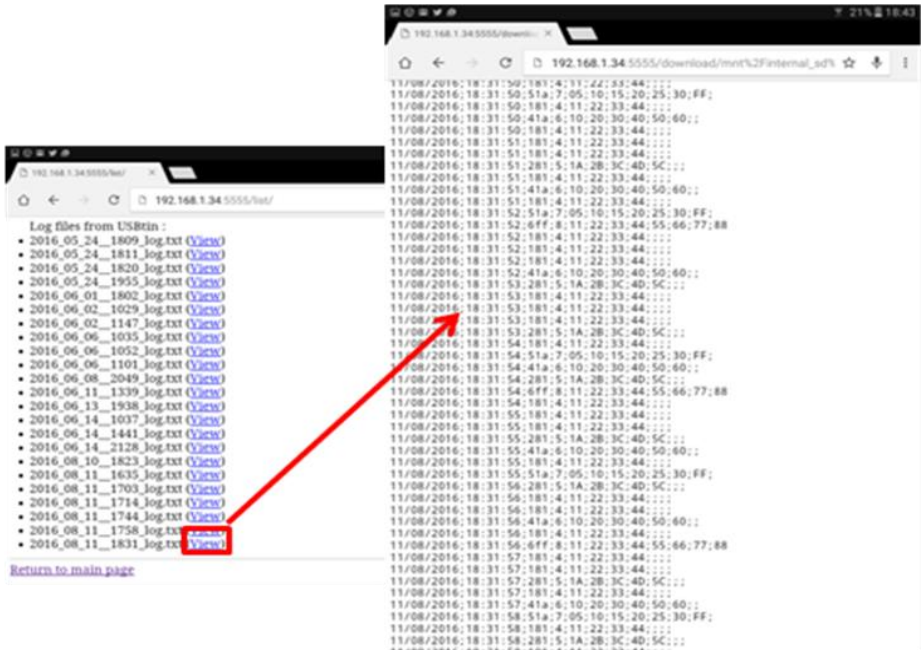
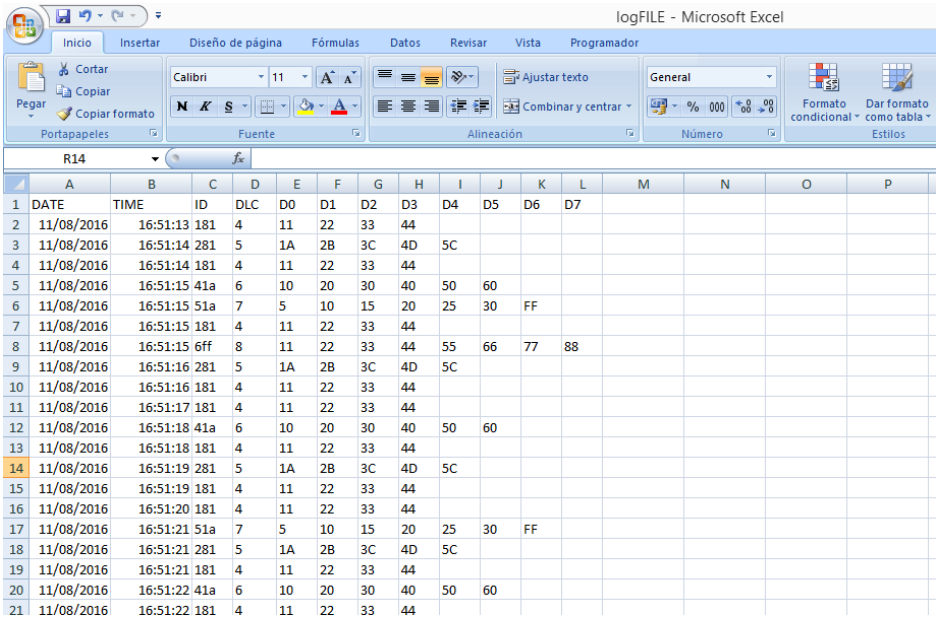


Figura 7-15. Consulta de la llista de fitxers d'històric de dades i visualització del contingut. Font pròpia

Com s'ha vingut avançant en diferents punts de la memòria, es vol estandarditzar el format dels fitxers de dades que es generen i s'opta per un fitxer en que separa els camps per un limitador ";", amb el que es coneix popularment com el format *CSV*. És per això que en la validació de la part del servidor web es descarrega un dels fitxers de dades disponible i s'obre amb l'editor Microsoft Office Excel, per veure si s'ha exportat correctament la informació, obtenint un resultat satisfactori, tal i com es pot comprovar en la Figura 7-16.



	A	B	C	D	E	F	G	H	I	J	K	L
1	DATE	TIME	ID	DLC	D0	D1	D2	D3	D4	D5	D6	D7
2	11/08/2016	16:51:13	181	4	11	22	33	44				
3	11/08/2016	16:51:14	281	5	1A	2B	3C	4D	5C			
4	11/08/2016	16:51:14	181	4	11	22	33	44				
5	11/08/2016	16:51:15	41a	6	10	20	30	40	50	60		
6	11/08/2016	16:51:15	51a	7	5	10	15	20	25	30	FF	
7	11/08/2016	16:51:15	181	4	11	22	33	44				
8	11/08/2016	16:51:15	6ff	8	11	22	33	44	55	66	77	88
9	11/08/2016	16:51:16	281	5	1A	2B	3C	4D	5C			
10	11/08/2016	16:51:16	181	4	11	22	33	44				
11	11/08/2016	16:51:17	181	4	11	22	33	44				
12	11/08/2016	16:51:18	41a	6	10	20	30	40	50	60		
13	11/08/2016	16:51:18	181	4	11	22	33	44				
14	11/08/2016	16:51:19	281	5	1A	2B	3C	4D	5C			
15	11/08/2016	16:51:19	181	4	11	22	33	44				
16	11/08/2016	16:51:20	181	4	11	22	33	44				
17	11/08/2016	16:51:21	51a	7	5	10	15	20	25	30	FF	
18	11/08/2016	16:51:21	281	5	1A	2B	3C	4D	5C			
19	11/08/2016	16:51:21	181	4	11	22	33	44				
20	11/08/2016	16:51:22	41a	6	10	20	30	40	50	60		
21	11/08/2016	16:51:22	181	4	11	22	33	44				

Figura 7-16. Resultat de l'exportació d'un fitxer de dades de l'aplicació amb format CSV. Font pròpia

Un cop executada la validació del servidor web es pot afirmar que ha estat satisfactòria, ja que s'aconsegueix interactuar amb l'estat del port CAN del hardware USBtin a través del servidor, es pot veure en temps real els missatges CAN que rep el dispositiu sense la necessitat d'estar connectats físicament a ell, i el servidor web permet la visualització i descarrega dels fitxers que contenen l'històric dels missatges rebuts.

8. Planificació temporal

Tot projecte requereix d'una planificació a nivell temporal per tal d'executar les diferents fases que el conformen de forma ordenada i complint els terminis d'execució. És per aquest motiu que en la fase prèvia a la realització d'aquest projecte l'autor va acordar conjuntament amb el tutor quines eren les fites i tasques generals que s'havien de realitzar per tal de poder portar a terme l'execució del projecte.

A continuació es llisten les tasques generals que l'autor ha portat a terme al llarg de la realització d'aquest projecte i s'adjunta la planificació temporal que s'ha seguit per executar-les.

I. Reunió inicial amb el tutor del projecte

L'autor del projecte té clar que vol desenvolupar una aplicació com a treball de final dels estudis del Màster i es posa en contacte amb el tutor Manuel Moreno Eguilaz per adquirir informació referent als treballs proposats des del Departament d'Enginyeria Electrònica de l'ETSEIB.

II. Recerca d'informació prèvia

Després de la proposta inicial sorgida de la reunió l'autor del projecte, autèntic desconegut de la plataforma de programació d'aplicacions B4A i del hardware que s'ha d'utilitzar, fa una recerca d'informació en fòrums de la plataforma, pàgines web on consultar tutorials de programació tant d'aplicacions Android com de llenguatge HTML, recerca d'informació del protocol de comunicacions HTTP, del funcionament de servidors web, entre altres ítems necessaris per l'execució de l'aplicació requerida en el projecte.

III. Proves de programació amb B4A

Es tracta d'un període d'iniciació amb el software de programació per adquirir els coneixements necessaris per dur a terme les tasques posteriors de desenvolupament de l'aplicació.

IV. Definició dels requeriments del projecte

Un cop dut a terme l'estudi de l'art per veure quins productes hi ha actualment similars al que es vol desenvolupar, i conegut el funcionament del software de programació, es defineix quin producte es desenvoluparà i quines prestacions tindrà l'aplicació CAN-Server.

V. Desenvolupament funcionalitats a nivell local

En aquesta primera fase d'execució es desenvolupa totes les parts de l'aplicació CAN-Server a nivell local.

VI. Desenvolupament funcionalitats remotes

En aquesta segona fase d'execució es desenvolupa el servidor web que emmagatzema l'aplicació, per dur a terme les funcionalitats locals de forma remota i interactuar amb el hardware a través d'altres dispositius connectats en la mateixa xarxa Wi-Fi.

VII. Test i validació de l'aplicació

Un cop finalitzada l'execució de les tasques V i VI es procedeix a la validació de les funcionalitats de l'aplicació per veure el seu correcte funcionament. Es tracta d'un procés executat després de la programació de l'aplicació, tot i que durant les fases de desenvolupament de forma paral·lela es testeja l'aplicació.

VIII. Redacció memòria del projecte

Un cop finalitzat el procés de desenvolupament es necessari redactar la documentació associada que reculli tot el treball desenvolupat per part de l'autor.

IX. Entrega de la documentació del projecte

Tràmit administratiu per tal d'acomplir amb la realització del projecte i que aquest sigui avaluat pel tribunal corresponent en la data fixada per l'escola.

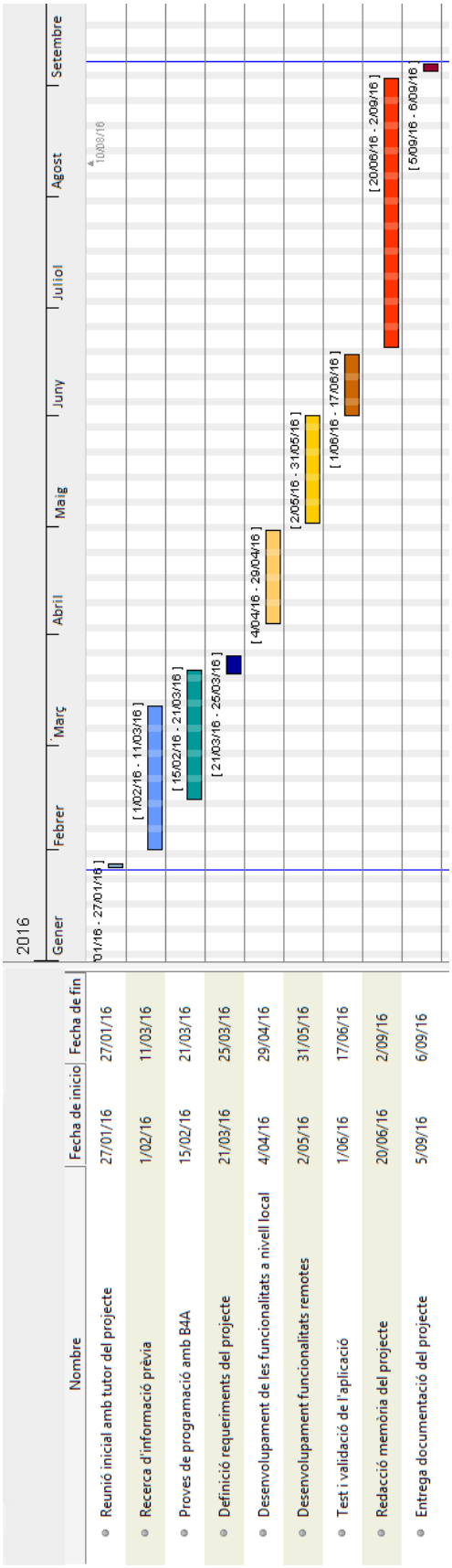


Figura 8-17:Planificació temporal d'execució de les fases del projecte. Font pròpia.

9. Pressupost del projecte

La valoració econòmica del projecte es pot consultar en la següent taula adjuntada en aquest apartat de la memòria.

CONCEPTE	Hores/ Quantitat	Preu unitari	COST
Resum de conceptes desenvolupament d'enginyeria			
Estudi de viabilitat	60 h	0,00€/h	0,00 €
Desenvolupament de l'aplicació	280 h	40,00€/h	11.200,00 €
Test i validació	80 h	40,00€/h	3.200,00 €
Redacció documentació	100 h	25,00 €/h	2.500,00 €
Resum de conceptes material i llicències			
Llicència B4A	1	66,85 €	66,85 €
Tableta Wolder Chicago 10''	1	74,90 €	74,90 €
USBtin	2	34,90 €	69,8 €
TOTAL:			17.111,55 €

El pressupost total del projecte, després de contemplar totes les hores d'enginyeria de desenvolupament de l'aplicació com el material utilitzat al llarg del projecte, ascendeix a 17.111,55 €

**NOTA: Preus del material utilitzat en el projecte consultats a la pàgina web [1] i [11]. El preu de la llicència és en el moment de l'adquisició del software per part l'autor del projecte.*

10. Línies futures d'investigació i implementació

Arribats en aquest punt de la memòria es bo fer un anàlisi del que no s'ha implementat en aquest projecte, ja que no estava dins del abast però que pot esser objecte d'estudi en un futur en altres projectes que s'impulsin des del Departament d'Enginyeria Electrònica.

L'eina desenvolupada per l'autor del projecte val a dir que es tracta bàsicament d'una eina amb permisos de lectura respecte al hardware utilitzat, és a dir, permet visualitzar que li està passant al dispositiu, quines dades està rebent, permet veure tots els missatges CAN que circulen per la xarxa de comunicacions a la que es connecta però no té drets d'escriptura, és a dir, l'aplicació no permet enviar missatges per la xarxa de comunicacions CAN.

Així doncs, una possible ampliació en un futur de l'aplicació seria la **d'incloure permisos d'escriptura al hardware USBtin**. És a dir, permetre l'enviament de missatges CAN des de l'aplicació,tant a nivell local com de forma remota a través del servidor web.

Una altra línia d'implementació futura en l'aplicació CAN Server pot ser la de **millorar la visualització *online* dels missatges que rep el dispositiu USBtin** i que fins al moment es presenten en forma de taula, tant en la tableta com en el servidor web. Pot estudiar-se el funcionament dels *sniffers* consultats en l'estat de l'art d'aquesta memòria per veure altres tècniques alhora de mostrar els missatges rebuts de forma instantània.

Una altra millora que es pot dur terme respecte a l'aplicació CAN Server és augmentar el nivell de seguretat per accedir al servidor web que allotja l'aplicació. El nivell de seguretat implementat en aquest projecte és baix, per la qual cosa es pot plantejar la **millora de la seguretat alhora d'accedir al servidor web**.

Així doncs, l'autor del projecte proposa tres línies de millora per tal de que en un futur es pugui ampliar les funcionalitats i les prestacions de l'aplicació CAN Server:

- ✓ Permetre l'enviament de missatges CAN des de l'aplicació tant a nivell local com a nivell remot a través del servidor web.
- ✓ Millorar la visualització *online* dels missatges CAN.
- ✓ Incrementar el nivell de seguretat en l'accés al servidor web.

Conclusions

Un cop arribat en aquest apartat, cal valorar els treballs realitzats per extreure les conclusions que se n'han derivat. Amb la realització d'aquest treball s'ha desenvolupat una eina de diagnòstic per a xarxes de comunicació que incorporen el protocol CAN per tal de que els usuaris puguin veure el tràfic de dades que circula per la xarxa.

Els estudis previs de mercat realitzats per analitzar productes similars al desenvolupat en aquest projecte van determinar que aquest tipus d'eines estan constituïdes per una solució hardware que permet dur a terme la connexió física amb la xarxa de comunicacions i una solució software que permet veure les dades que s'hi transmeten.

Així doncs, després d'analitzar el mercat es proposa utilitzar el hardware USBtin per comunicar-se amb la xarxa CAN i a través de la plataforma B4A es crea una aplicació capaç de córrer en dispositius Android. D'aquesta forma, s'optimitza els recursos dels que es disposa al Departament d'Enginyeria Electrònica, tant a nivell d'eines com de documentació d'anteriors projectes.

Els treballs de la fase de desenvolupament s'han centrat en la implementació de l'aplicació Android, que divideix la seva arquitectura en dos nivells. Una capa local que permet la comunicació amb la xarxa a través del dispositiu on s'instal·la l'aplicació, i una altra capa que allotja un servidor web i que permet la interacció remota amb el hardware USBtin.

Per últim, els treballs realitzats en la fase de test i validació han permès veure i corregir elements de l'aplicació per aconseguir que el sistema desenvolupat funcioni de la forma prevista i definida en els requeriments de l'aplicació.

Agraïments

L'autor del projecte vol expressar en primer lloc l'agraïment al tutor del projecte Manuel Moreno Eguílaz per l'amabilitat i la disponibilitat mostrada en tot moment de la realització del projecte.

En segon lloc, l'autor del projecte també vol agrair a la seva novia Laura, la icona que va dissenyar per tal d'incorporar-la com a logotip de l'aplicació.

Bibliografia

Referències bibliogràfiques

[1] Tomas Fischl, "USBtin - USB to CAN interface", 2016, [Online]. Disponible: http://www.fischl.de/usbtin

[2] Marcel Garrido, "Interfaz Gráfica de usuario usando una tableta Android y un adaptador CAN/USB", UPC, 2015.

[3] Héctor Hernández, "Caracterización de ciclos de conducción utilizando una tableta Android y bus CAN", UPC, 2016.

[4] Peak Systems,"PCAN-USB - CAN Interface for USB", 2016, [Online]. Disponible: http://www.peak-system.com/PCAN-USB.199.0.html?&L=1

[5] CAN in Automation CiA, 2016, [Online]. Disponible: http://www.can-cia.org/

[6] Gwentech Embedded, "Android CAN Bus Bluetooth Adapter", 2016, [Online]. Disponible: http://gwentechembedded.com/products/gt1027-android-can-bus-bluetooth-adapter/#

[7] ICPDAS USA, "I-7540D-WF CAN to Wi-Fi converter", 2016, [Online]. Disponible: http://www.icpdas-usa.com/products.php?PID=4013#reference

[8] Anywhere Software, "B4A Comunity", 2016, [Online]. Disponible: https://www.b4x.com/android/forum/

[9] W3Schools, "HTML Tutorial", 2016, [Online]. Disponible: http://www.w3schools.com/html/

[10] Anywhere Software, "B4A Library [Class] Flexible Table", 2016, [Online]. Disponible: https://www.b4x.com/android/forum/threads/class-flexible-table.30649/

[11] Wolder Electronics, "miTab CHICAGO", 2016, [Online]. Disponible: http://wolderelectronics.com/productos/177-mitab-chicago#.V7825_mLTIU